18th

$15.00

# ARRL and TAPR DIGITAL COMMUNICATIONS CONFERENCE

PHOENIX, ARIZONA

**September 24-26, 1999**

# 18th ARRL and TAPR DIGITAL COMMUNICATIONS CONFERENCE

**Co-Hosts:** Motorola Amateur Radio Club of Arizona (MARCA)
Packet Radio Users Group of Japan (PRUG)


**ARRL**
225 Main Street
Newington, CT 06111-1494 USA
tel: 860-594-0200 http://www.arrl.org

Tucson Amateur Packet Radio
8987-309 E. Tanque Verde Rd #337
Tucson, Arizona 85749-9399 USA
tel: 940-383-0000 http://www.tapr.org

Printed in USA



ISBN: 0-87259-767-9

ARRL Order Number: 7679

First Edition

# Welcome!

We live in an increasingly digital world. Military and commercial telecommunications have embraced digital technology with gusto. With the proliferation of digital consumer electronics—from portable MP3 players to DVD-based home theaters—the public has also become more comfortable with the "digital reality." This trend will only accelerate in the near future.

So where does Amateur Radio stand?

With the exception of certain DSP applications and innovations such as APRS and PSK31, most hams still have both feet firmly planted in the analog world. If we expect to move forward with the rest of telecommunications community, we must do more.

The Eighteenth ARRL and TAPR Digital Communications Conference offers a valuable opportunity to expand the amateur digital "envelope." As you explore these proceedings you'll find fascinating applications of amateur digital technologies. These authors are true innovators, doing their utmost to ensure that Amateur Radio isn't left behind in the digital revolution.

Read the proposals and apply the ideas. Or, use the ideas to inspire your own creative passions. Perhaps we'll have the pleasure of reading one of your articles in next year's edition!

David Sumner, K1ZZ
ARRL Executive Vice President

September 1999

# Table of Contents

# HAM WEB NODE

Dale Puckett, K0HYD, and John Bandy, W0UT
23440 W. Highway 54, Goddard, Kansas 67052 U.S.A.
k0hyd@feist.com; john.bandy@twsubbs.twsu.edu

**Abstract**

This paper describes an existing amateur packet node that provides many graphic and text services to other amateur packet stations in the amateur radio 2 meter band.

**Keywords**

Amateur radio, packet, http, web node, graphics, digital, Linux, VHF.

**Introduction**

Packet radio activity in Kansas has declined dramatically in the past several years. This trend which has also shown up in the rest of the nation, can be partially contributed to the immense popularity of the internet and the migration of digital operators to the World Wide Web. The goal of this project is to reverse this trend and bring amateur radio operators interested in the digital domain back to the packet radio fold.

Because of the easy access to commercial web browsers and the popularity of the http medium, we reasoned that an amateur packet node running at 9600 baud would provide an attractive entrance into the http world, if we kept the graphics load relatively low. This concept was proved in practice and we have been able to download a simple html page with several graphics (including a 24K image of a QSL card) in 26 seconds. We have also used the ftp protocol to transfer several large documents each day.

To test the theory, we initially set up two stations -- a Linux based node operating on three frequencies and a client operating on the 9600 baud LAN (145.63 MHz). Both operators use a graphic user interface (GUI) running under Linux. The Web Node operator uses the KDE desktop and the client operator uses X Windows. The long term goal is to let amateur operators running Windows 98 use the TCP/IP protocol over ax25 to communicate with the Web Node. They can then access the ham web node by requesting the nodes URL in their favorite browser. For example, the client station now connects to the node by entering "http://k0hyd.ampr.org/" in his browser address field.

**Infrastructure**

The K0HYD node operates on three frequencies. The 9600 baud LAN where this experimentation is taking place is on 145.63 MHz. A 1200 baud, 145.01 node provides a link to Kansas packet operators to the West and South of Wichita. A 9600 baud node on 430.55 MHz provides operators on the 9600 LAN access with a high speed connection to Topeka and Kansas City on Kansas Packet Radio backbone. While most TCP/IP operation is conducted on 145.63 MHz, TCP/IP routing has been set up over NetROM on the backbone with BNSBB, N0RLR, in Burlington, KS. This allows routing to both Northeast and Southeast Kansas.

NetROM nodes are available on all three frequencies. However, the full node list is only transmitted on the local 9600 baud LAN. In addition to the NetROM services, a full service F6FBB bulletin board is available to operators on all three frequencies. A simple DX Cluster, presently DXNet, is also available on all three nodes. Eventually, the cluster will operate on 145.97 MHz with a tie to neighboring clusters on 430.55 MHz. We are working with DX operators and officers of the Wichita Amateur Radio Club (WARC) to determine the interest level of this service. If interest dictates, we also hope to provide a gateway to the internet in the future.

The remainder of this paper will detail the services provided and list the configuration files needed by a Linux AX25 operator.

**Services**

Web pages: HTTP via TCP/IP. /http://k0hyd.ampr.org/ [44.122.0.4].
FTP using TCP/IP: 9600 bps download/upload of graphic files, etc.
Telnet using TCP/IP: Amateurs may run software on host server.
Mail with SMTP and POP3 servers using TCP/IP.
News using TCP/IP.
NetROM Nodes via AX25.
DX Cluster via AX25.
Public Bulletin Board System (F6FBB) via AX25.
All Standard AX25 services.

**Hardware**

Cyrix P133+ based computer
Four (04 ) Serial Ports
Kantronics 9612 TNC
Kantronics 9612 Plus TNC
Motorola Mitrek running 50 watts on 145.63 MHz
Yaseu FT-5100 running 50 watts on 145.01 MHz
Tekk KS-1000 on 430.55 MHz
Mirage BD-35 dual band amplifier with 32+ watts out
Comet GP-9 dual band antenna (02)
Duplexer
FLEXI 4XL and RG-213 Coax

**Software**

Caldera OpenLinux 2.2 operating system
KDE 1.1.1 desktop for Linux
Apache Web Server
F5MZN DxCluster
XFBB F6FBB X Window based Bulletin Board System (PBBS)
LINUX AX25 Kernel Support
AX25 Utilities
sunsite.unc.edu/pub/Linux/.
www.us.kernel.org
www.kde.org

**Technical Support**

Nate Bargmann, N0NB
Caldera Systems  www.calderasystems.com
Majordomo@vger.rutgers.edu
  subscribe linux-hams
  subscribe linux-newbie

**Helpful books and journals.**

Linux AX25-HOWTO Amateur Radio, Terry Dawson, VK2KTJ
OpenLinux User's Guide
ARRL Handbook
Linux Journal
QST Journal
TAPR Packet Status Register

**Software Configuration Details**

Configuring a full service Linux-hosted node is not for the faint of heart. There are what seems like dozens of configuration files, used by dozens of applications, many of them running in the background as daemons. Yet, if you approach the problem in a logical fashion and do a lot of reading, the plot thickens and everything comes together. The secret is to gather all the information you need to do the job.

In this section we'll attempt to show you the information we gathered together to assemble the full service node described above. By following the examples below and changing the names, callsigns and ip addresses to protect the innocent, you will find it relatively easy to get a Linux station on the air.

We'll take a look at the **axports** file first.

```
# /etc/ax25/axports
#
# The format of this file is:
#
# name callsign speed paclen window description
#
01     K0HYD-13      19200  255    4      145.01 MHz (1200  bps)
63     K0HYD-12      19200  255    4      145.63 MHz (9600 bps)
55     K0HYD-9       19200  255    4      430.55 (9600 KS State Backbone)
p1     K0HYD-11      19200  255    4      Pipe from Node
p2     K0HYD-10      19200  255    4      Pipe to BBS
```

The first thing you'll notice is the fact that each physical port on a Linux AX25 system has its own callsign. If I call another station on 145.01, the system uses the callsign, K0HYD-13. The other callsigns in the table above are used when a station is called on the frequency described in their row. The 19200 is the speed of the rs-232 connection between the computer and the tnc. The ports p1 and p2 are special. They represent internal pipes within Linux. p1 takes the data from the node. p2 receives that data for the F6FBB BBS.

We used the high SSID numbers for these internal ports because they are not generally used by the end user. They are only used by the Linux system to communicate with other stations. End users will use the NetROM nodes which generally have lower SSID numbers.

Speaking of NetROM, we'll look at the **nrports** file next.

```
# /etc/ax25/nrports
#
# The format of this file is:
#
# name callsign alias paclen description
#
NetROM K0HYD-14      #ICTSW 235    NetROM Switch
netnod K0HYD-1       KSICT  235    Node Console
netbbs K0HYD-3       ICTBBS 235    FBB Full Service BBS
netdx        K0HYD-5       ICTDX  235    Experimental DX Cluster
```

The names in the first column of this table are the internal names of the four NetROM nodes used at K0HYD. They each use the callsigns in the second column of their row. The name of the nodes broadcast to neighboring NetROM nodes are listed in the third column. The fourth column contains the number of bytes in each packet and the last column contains a short description of each node. Notice here that the SSIDs on these callsigns is in the lower range most commonly expected by end users.

Notice here that the node named NetROM is defined as a hidden node by using a pound sign at the beginning of the name. While end users can connect to #ICTSW, they can't do anything else once connected. For this reason, this node is hidden from view. To connect to the node and travel on up the backbone, end users must connect to the Node Console, KSICT. The nrports file works with the **nrbroadcast** file to set up your NetROM nodes.

```
# /etc/ax25/nrbroadcast
#
# The format of this file is:
#
# ax25_name min_obs def_qual worst_qual verbose
#
01     5      190    100    0
63     5      192    70     1
55     5      192    100    0
```

Notice here that the quality for each physical port defined in the axports file may be assigned individually. By placing the def_qual of the 55 port higher than the 01 port, the operator forces the station to attempt a connection on the high speed port before trying on the low speed port. This strategy is required when the same node is available on more than one frequency. If the first connection fails, the node will attempt a connection on the other frequency. Take special notice also of the 0 and 1 in the last column. This column is telling us that the node broadcasts on the 145.63 MHz 9600 baud LAN are verbose. All the nodes available to KSICT are made available to the end users on this frequency. Conversely, the other nodes on the 1200 and 9600 baud backbones do not need this information so they are not set to verbose. Using the non-verbose mode prevents the broadcast of weak, almost non-existent links that are created when a short band opening delivers a node broadcast to the local node.

How does the Linux system know what program to run when another station connects? By using the **ax25d.conf** file. We've abbreviated the listing to save space.

```
# /etc/ax25/ax25d.conf
#
```

```
# ax25d Configuration File.
#
# AX.25 Ports begin with a '['.
#
#
[K0HYD-0 VIA 01]
NOCALL   * * * * * *  L
default  * * * * * *  -          root  /usr/local/sbin/ttylinkd   ttylinkd
#
[K0HYD-0 VIA 63]
NOCALL   * * * * * *  L
default  * * * * * *  -          root  /usr/local/sbin/ttylinkd   ttylinkd
#
[K0HYD-0 VIA 55]
NOCALL   * * * * * *  L
default  * * * * * *  -          root  /usr/local/sbin/ttylinkd   ttylinkd
#
[K0HYD-1 VIA 01]
NOCALL   * * * * * *  L
default  * * * * * *  -          root  /usr/local/sbin/node node
#
[K0HYD-7 VIA 63]
NOCALL   * * * * * *  L
default  * * * * * *  -          root  /usr/local/sbin/axspawn    axspawn %u +
#
[KSICT VIA 55]
NOCALL   * * * * * *  L
default  * * * * * *  -   root  /usr/sbin/node      node
#
# NET/ROM Ports begin with a '<'.
#
<netnod>
NOCALL   * * * * * *  L
default  * * * * * *  -   root  /usr/sbin/node      node
#
#<NetROM>
#NOCALL  * * * * * *  L
#default * * * * * *  -   root  /usr/sbin/node      node
```

The first three entries above tell the Linux software that any station connecting to K0HYD-0 on any one of the three ports will be connected to the ttylinkd daemon which provides a method for chatting keyboard to keyboard. Actually, you repeat each callsign for each port you want to listen on. We have deleted the duplicate entries here to save space.

The next entry shows that any station connecting to K0HYD-7 on the 145.63 MHz port will be connected to the axspawn program. This AX25 Linux Application program spawns a Linux shell for the connected station and lets the operator drive the Linux computer.

The next line shows that the system will listen for the node name KSICT on the 430.55 MHz port. When it hears a station, it will connect it to the node program. This program emulates the familiar interface users expect when the connect to a NetROM or theNet node.

Notice that only the netnod and NetROM ports are used in the ax25d.conf file. The others are left out of this file to prevent the ambiguity caused by the fact that the XFBB BBS software is also listening for these nodes on the same physical hardware port. The XFBB and the DxNet software use the nodes named ICTBBS and ICTDX. Notice also that the callsign SSID used by the FBB BBS software, K0HYD-3, cannot be used by any other software. For this reason there are no listeners in the ax25d.conf file for K0HYD-3. Here is the operative part of the FBB **port.sys** file.

```
#
#TNC NbCh Com MultCh Pacln Maxfr NbFwd MxBloc M/P-Fwd Mode  Freq
 0    0    0   0      0     0     0     0      00/01   ----  File-fwd.
 1    8    2   63     250   2     1     10     30/60   XUWY  145.63
 2    2    2   01     250   4     1     10     00/60   XUWY  145.01
 3    6    2 netbbs   250   4     1     10     15/60   XUWY  NetROM
 4    6    2   p2     250   4     1     10     15/60   XUWY  ViaNode
#
```

In the fourth row of this table the FBB software is told to listen to the netbbs device. This NetROM device goes by the name of ICTBBS.

Here is the corresponding **DxNet config** file.

```
SET LANGuage english
SET/QTH Goddard, KS
SET/HOME ICTDX
SET/LOCAtor EM17EN
SET/call k0hyd-5
SET/ssid +5
SET/port 01 55 63 netdx
```

Notice here that this software presently listens on all three ax25 ports and on the netdx port which goes by the node name of ICTDX.

To connect to the BBS software when connected to the KSICT node, the user issues the BBS command. It is set up in the **node.conf** file. A few of the key lines are listed here.

```
HostName     k0hyd.ampr.org
LocalNet     44.122.0.0/24
Alias          BBs      "c p1 k0hyd-3"
NodeId       #ICTSW:K0HYD-14
NrPort       NetROM
```

The third line above shows that when a node user types "bb" the Linux system will use the p1 port, a Linux pipe defined in axports, to connect him to K0HYD-3 which is the only callsign the BBS uses.

**Getting it all started**

If you started up your Linux based amateur station manually, it could turn into quite a job, require a great memory and take a lot of time. The solution to this dilemma is to use a Linux shell script. We'll highlight the command needed to bring your station on the air in the listing below.

```
#!/bin/sh
# /etc/ax25/start
# This file starts the ax25 Programs
# First, the NetROM device must be brought down so they may be created again
echo "Beginning the ax25 setup ..."
echo "Erase the ifconfig nr0, nr1, nr2 and nr3 entries ..."
ifconfig nr0 down
ifconfig nr1 down
ifconfig nr2 down
ifconfig nr3 down
# Next, you must attach the ax.25 KISS interface to ttyS1
# We are using two, dual port TNCs at this station.
# Two pseudo ports, one for each TNC channel are created first.
# The first TNC is on ttyS1 (COM2)
```

```
echo "Attaching ax.25 interfaces ... KPC-9612"
mkiss -s 19200 /dev/ttyS1 /dev/ptyq0 /dev/ptyq1
sleep 2
# Then, we attach each port.
kissattach -i 44.122.0.4 -m 512 /dev/ttyq0 01
kissattach -i 44.122.0.4 -m 512 /dev/ttyq1 63
# And, set the desired operating parameters.
kissparms -p 01 -t 250 -s 200 -r 25
kissparms -p 63 -t 90 -s 90 -r 25
# The second TNC is on ttyS0 (COM1)
# Attach the ax.25 KISS interface to ttyS0
echo "Attaching ax.25 interfaces ... KPC-9612 Plus"
mkiss -s 19200 /dev/ttyS0 /dev/ptyq2 /dev/ptyq3
sleep 2
# kissattach -i 44.122.0.4 -m 512 /dev/ttyq2 XX
kissattach -i 44.122.0.4 -m 512 /dev/ttyq3 55
# kissparms -p XX -t 250 -s 200 -r 25
kissparms -p 55 -t 90 -s 90 -r 25
# Now set up Alias from BBS through Node
# We use a Linux pipe to accomplish this.
sleep 3
kissnetd /dev/ptys1 /dev/ptys2 &
sleep 3
kissattach -i 44.122.0.4 -m 512 /dev/ttys1 p1
kissattach -i 44.122.0.4 -m 512 /dev/ttys2 p2
# Configure the IP Routing over AX25
echo "Now, configure the IP Routing ..."
route add -net 44.122.0.0 netmask 255.255.255.0 ax0
route add default ax0
route add pc.k0hyd eth0
ifconfig ax0 44.122.0.4 hw ax25 k0hyd-13
ifconfig ax0 broadcast 44.122.0.255 netmask 255.255.255.0
ifconfig ax1 44.122.0.4 hw ax25 k0hyd-12
ifconfig ax1 broadcast 44.122.0.255 netmask 255.255.255.0
ifconfig ax2 44.122.0.4 hw ax25 k0hyd-9
ifconfig ax2 broadcast 44.122.0.255 netmask 255.255.255.0
ifconfig eth0 broadcast 44.122.0.255 netmask 255.255.255.0
echo "Start All Services ..."
sleep 3
# Configure NetROM with TCPIP
nrattach -i 44.122.0.4    NetROM
nrattach -i 44.122.0.4    netnod
nrattach -i 44.122.0.4    netbbs
nrattach -i 44.122.0.4  netdx
ifconfig nr0 broadcast 44.122.0.255
ifconfig nr1 broadcast 44.122.0.255
ifconfig nr2 broadcast 44.122.0.255
ifconfig nr3 broadcast 44.122.0.255
# We have one preferred neighbor on the backbone
# The route to him is locked.
echo "Form a locked route table ..."
nrparms -routes 55 n0kta-1 + 198
sleep 3
# Configure Routes to Individual TCPIP Stations
echo "Configure routes to TCPIP stations ..."
arp -H ax25 -s 44.122.0.2 wf0a-10
arp -H ax25 -s linux.n0nb n0nb-8
arp -t NetROM -s n0rlr ka0oxh-7
nrparms -nodes n0rlr-2 + BNSBB 192 6 55 n0kta-1
route add n0rlr nr0
route add wf0a ax0
route add w0ut ax1
```

```
sleep 4
# Now that most of the configuration has been initialized,
# we must start all the daemons that run in the background
ax25d &
sleep 3
netromd &
sleep 3
mheardd &
sleep 2
echo "Load the old Node Table"
/root/nodes.save
echo "Associating Users to callsigns ..."
axparms -assoc w0ut w0ut
axparms -assoc n0kta n0kta
echo "Now issue beacon to tell them we're here ..."
beacon 01 -d CQ -t 15 "KSICT [K0HYD-1], Full service Linux NetROM/TCPIP node
in Goddard"& beacon 63 -d CQ -t 15 "KSICT [K0HYD-1], Full service Linux
NetROM/TCPIP node in Goddard"&
beacon 63 -d CQ -t 15 "KSICT [K0HYD-1], Full service Linux NetROM/TCPIP node
in Goddard"& beacon 63 -d CQ -t 15 "KSICT [K0HYD-1], Full service Linux
NetROM/TCPIP node in Goddard"&
sleep 5
echo "All AX25 services should now be configured."
```

All of the configuration files used at K0HYD.AMPR.ORG are being made available for http download at http://www.fn.net/~k0hyd

**Acknowledgements.**

The authors are grateful to Nate Bargmann, N0NB, who built the first Linux AX25 Node in our community and converted the surplus Motorola Mitrek VHF mobile transceiver to the 9600 bps data radio.

We were also assisted by Joe Johnson, N0KTA, Richard Rose, WF0A, and others.

**Reference.**

[1] Jeff Tranter, VE3ICH, "Packet Radio Under Linux", Linux Journal, SSC, Seattle, WA, Issue #41, pp.44-47, September 1997, phone 206-782-7733.

[2] Fred Treasure, "NF/Observatory Networking with Linux OS", Linux Journal, SSC, Seattle, WA, Issue #34, pp. 14-17, February 1997, phone 206-782-7733.

[3] Dale Puckett, K0HYD and John Bandy, W0UT, Ham Web Station, Submitted to 18th ARRL and TAPR Digital Communications Conference, Phoenix, AZ.

(4) MVFMA AX.25 Configuration Files at http://www.febo.com/packetnet/mvfma.html

**Trademarks.**

Linux is not a trademark, and is not connected to UNIX or X/Open. The X Window System is a registered trademark of the Massachusetts Institute of Technology.

**No Warranty.**

This material is provided "as is" and without any expressed or implied warranties, including, without limitations, the implied warranties of merchantability and fitness for a particular purpose.

Downloadable from http://www.fn.net/~k0hyd/ax25server.tar.gz

**Ham Web Station** August 9, 1999

John Bandy, W0UT and Dale Puckett, K0HYD
2810 Euclid
Wichita, Kansas 67217-1927 U.S.A.
john.bandy@twsubbs.twsu.edu
k0hyd@feist.com

**Abstract.**

This paper explains the requirements for an amateur radio station that communicates with an amateur radio web station/node on the 2 meter band. It uses a graphic user interface (GUI) desktop of a personal computer (PC).

**Keywords.**

Amateur radio, packet, web station, graphics, digital, Linux, VHF

**Introduction.**

Packet amateur radio [4] has advanced to the point it is now possible to build a packet station that processes web pages over amateur radio VHF frequencies. The building blocks are free or can be purchased at reasonable prices. Support systems are in place to make it happen. This paper will describe an actual working client station.

This beginner's station is being used as a start toward building a full blown ham radio web server station. It will teach network administering and PC technical supporting. The homebrewing and experimenting has been a joy.

**Hardware.**

This client station consists of the following hardware.

Homebrewed PC with an Intel 486DX2-66 cpu and 32 Meg..
Mitsumi Electonrics Corp Mouse ECM-S3101
Tigertronics BayPac BP_96A Modem in the PC parallel port.
MFJ VHF 5 watts Data Radio, FSK mode, 145.630 Mhz. [1]
Belden 9913 coax fitted with N connectors (50 feet).
Homebrewed WD4FAB 2 meter/440 Mhz J-pole up 20 feet. [2]
Hamtronics LNS-144 Kit mounted on the J-pole.
Homebrewed 13.8-V,5-A G4YNM power supply. [3]
Grounding strap and rod.
MFJ 2 Meter FM Signal Analyzer.
MFJ SWR Analyzer.
EICO model 566 VOM Meter.

**Software.**

The softwares for the this user station are:
Red Hat Linux 4.2 network operating system cd-roms.
X Window System (GUI) on Red Hat cd-roms.
ImageMagick on Red Hat cd-roms.
ax25-module-14c.tar.gz from ftp.pspt.fi/pub/linux/ham/ax25/.
ax25-utils-2.1.42a.tar.gz from ftp.pspt.fi/pub/Linux/ham/.
net-tools-1.33.tar.gz from ftp.inka.de/pub/comp/Linux/.
web client start up script. see appendix.
web client configuration files. see appendix.

**Technical Support.**

Technical support was available from the following sources.
Air Capital Linux Users Group.
Ham Web Server Station Chief Operator.
Ham Client Stations Chief Operators.
Red Hat Software, Inc. for 30 days.
Majordomo@vger.rutgers.edu
subscribe linux-hams
subscribe linux-newbie
National Computer Resources (PC and parts dealer)

**Helpful books and journals.**

NEDA 1994 Annual Report, North East Digital Association, P.O. Box 563, Manchester, NH 03105.

Linux Journal, SSC, Seattle, WA, phone 206-782-7733

AX25 HOWTO on Red Hat cd-rom.

Clock-mini HOWTO on Red Hat cd-rom.

The Official Red Hat Linux Users Guide, Red Hat Software, Inc., Durham, NC, 1997.

Ian Wade, G3NRW, NOSintro-TCP/IP Over Packet Radio, Dowermain Ltd, Luton, Bedfordshire, UK, 1992.

**Operating Hints.**

Before using the 30 watts station K0HYD located 15 miles away at 57 feet, a command is entered (ping -i6 -s255 -c100 -pff k0hyd). This causes it to acknowledge 100 transmissions from the 5 watts station W0UT at 20 feet, and displays the percentage of unacknowledged transmissions. This loss percentage indicates what the conditions of the 2 meter band are. Some of the results are:

| CDT Time | % Lost August 1999 | | | | | |
|---|---|---|---|---|---|---|
| 0001-0100 | | | | | | |
| 0101-0200 | | | | | | |
| 0201-0300 | | | | | | |
| 0301-0400 | | | | | | |
| 0401-0500 | | | | | | |
| 0501-0600 | | | | | | |
| 0601-0700 | 23 | | | | | Sunrises |
| 0701-0800 | 4 | 2 | | | | |
| 0801-0900 | 6 | 57 | | | | |
| 0901-1000 | 71 | 14 | | | | |
| 1001-1100 | 5 | | | | | |
| 1101-1200 | 15 | | | | | |
| 1201-1300 | 15 | 11 | | | | |
| 1301-1400 | 6 | | | | | |
| 1401-1500 | 8 | | | | | |
| 1501-1600 | 10 | 8 | 73 | 85 | 9 | |
| 1601-1700 | 73 | 24 | | | | |
| 1701-1800 | 5 | 5 | 35 | 35 | | |
| 1801-1900 | 71 | 3 | 11 | 91* | | |
| 1901-2000 | 85 | 21 | | | | |
| 2001-2100 | 42 | 34 | | | | Sunsets |
| 2101-2200 | 18 | 50 | | | | |
| 2201-2300 | | | | | | |
| 2301-0000 | | | | | | |

* - Thunderstorm between client & server.

A retired commercial radio technician advised that an increase in effective radiated power (ERP) of the client station would improve the loss percentages.

For instance, "ping" was done and the packets lost were 2%. A subsequent upload of a 149,970 bytes .zip file of paintings to the server located in Goddard, Kansas took 21 minutes at 9,600 bits per second (bps). This same file was downloaded from a twisted-pair BBS in 3 minutes, zero seconds (stopwatch) at 9,600 baud (zmodem). There are several bits in a baud. Thus 9,600 baud is several times faster than 9,600 bps.

Another time when the loss was 6%, a web page with a QSL card image was downloaded from the Apache server into a Red Baron browser at 26 seconds (stop watch).

Some command examples:

| | |
|---|---|
| ps -ax | displays processes running. |
| mheard | displays copied callsigns. |
| call bp-96a k0hyd-3 | ax25 connect to a pbbs. |
| finger w0ut@k0hyd.ampr.org | tcp/ip command to id a host user. |
| ftp k0hyd.ampr.org | tcp/ip command to start file upload/ download program on a host. |
| startx | starts The X Window System (desktop). |

| | |
|---|---|
| http://k0hyd.ampr.org | downloads the ham web page. |
| sethdlc bc0 -c 15 | sends a deviation calibrate signal. |
| route -ne | lists the TCP/IP routes. |
| netstat -ot | displays net statistics. |
| ifconfig | displays device configuration. |

**Building Hints.**

When installing the Tigertronics BayPac BP-96A modem in the parallel printer port, re-compile the kernel with the printer support[] and PLIP...[]. A message in the dmesg report of "lp1 at 0x378 (polling) means the printer device lp1 is at the parallel port instead of the modem device bc0.

Set all maxmium transmission unit (MTU) and packet length (paclen) parameters to 255 bytes.

Any kernel above 2.0.36 will not compile the ax25 utilities correctly. The earlier binaries will run on the later kernels.

Set the deviation pot on the modem so the transmitted calibrated signal is deviating 3.0-3.5 Khz per the deviation meter copying it.

Set the frequency of all stations with the same frequency counter. Different counters can be used if they have been recently certificated.

Use a preamp with the MFJ Data Radio. [1]

Use RG8 coax and keep it short. [5]

**Acknowledgements.**

The authors are grateful to Nate Bargmann, N0NB, who started this Linux amateur radio craze in the Wichita Metro area, and has continued to help us.

We were also assisted by Joe Johnson, N0KTA, Tom Willis of the Air Capital Users Group, Earl Russell, KB0WKU, and numerous others.

Thanks goes to Ellen Bandy for pointing out grammatical and composition anomalies.

**References.**

[1] Steve Ford, WB8IMY, Managing Editor, "Product Review: MFJ-8621 Packet-Only Data Radio", QST Journal, American Radio Relay League, Inc., Newington, CT, Vol. 80, No. 4, April 1996, pp. 73-74, phone 860-594-0200.

[2] Dick Jansson, WD4FAB, "Antennas For Microsat Ground Stations", Getting Started on Amateur Radio Satellites, The Amateur Satellite Corporation - North America (AMSAT-NA), Silver Spring, MD, pp. 2,3, & 5, phone 301-589-6062.

[3] "A 13.8-V, 5-A Power Supply", 1995 ARRL Handbook, American Radio Relay League, Inc., Newington, CT, Chapter 11, pp.28-30, 1994, phone 860-594-0200.

[4] Jeff Tranter, VE3ICH, "Packet Radio Under Linux", Linux Journal, SSC, Seattle, WA, Issue #41, pp.44-47, September 1997, phone 206-782-7733

[5] "Losses in Transmission Lines", 1999 ARRL Handbook, American Radio Relay League, Inc., Newington, CT, Chapter 19, pp. 19.5-19.8, 1998, phone 860-594-0200.

**Trademarks.**



**No Warranty.**



**Appendix A. AX25 Start-up script and configuration files.**

Downloadable from http://www.fn.net/-k0hyd/ax25user.tar.gz

```
#!/bin/sh
#This is /usr/local/sbin/start_ax25, a shell script to submit
#the /usr/local/sbin/ax25 shell script.  It also puts non-error
#messages in the "report" file, and advisory and error messages
#in the "reporte" file.
#
if [ -z $1 ]
then
echo "need a script name.  Example: ./start_ax25 ax25"
else
echo "starting"
./$1 2> reporte > report
fi
```

```
#! /bin/sh
# This is /usr/local/sbin/ax25, a shell script to start the AX.25 networking
# system on w0ut.ampr.org
#
# Configure the bc0 device driver as a Parallel port
# If "lp1 at 0x0378, (polling)" show in dmesg then re-compile kernel with
# parallel printer support [] and PLIP..[].
# "6 lp" should disappear from cat /proc/devices list.

sethdlc -p -i bc0 mode par96 io 0x378 irq 7

# Configure the bc0 device with TxDelay, SlotTime, PPersist, ..plex.

sethdlc -i bc0 -a txd 200 slot 100 ppersist 40 half

# Configure the bc0 device with the ax25 callsign & bring it up (active/enable).

#/sbin/ifconfig bc0 hw ax25 bp-96a up

# Configure the bc0 device with the ax25 callsign

/usr/sbin/axparms -setcall bc0 w0ut-8

# Bring bc0 device up (active/enable).

/sbin/ifconfig bc0 up

# Setup TCP/IP datagram routing on Port bp-96a

/sbin/ifconfig bc0 w0ut broadcast 44.122.0.255 netmask 255.255.255.0 mtu 255 up
# /sbin/route add 44.122.0.0 netmask 255.255.255.0 bc0

# Start the ax25 daemon

/usr/sbin/ax25d

# Start the MHeard daemon

/usr/sbin/mheardd -n 20

# Associate W0UT with user jbandy
/usr/sbin/axparms -assoc w0ut jbandy

# Setup "loopback" to this host "w0ut" through bc0
```

```
/sbin/route add w0ut bc0

# Add K0HYD to ip routing table

/sbin/route add k0hyd bc0

# Display interrupts

cat /proc/interrupts

# Display devices in use

cat /proc/devices
```

```
# /etc/ax25/axports
# AX25 Port descriptions.
# The format of this file is:
#
# name  callsign      speed paclen window    description
#
bp-96a  W0UT-8        9600    255     1   145.630 Mhz (9600 bps) BayPac par
```

```
# /etc/ax25/ax25d.conf
#
# ax25d Configuration File.
#
# AX.25 Ports begin with a '['.
#
# All ports must be commented out except the ones to be used.
#
[W0UT-8 via bp-96a]
NOCALL   * * * * * *  L
default  * * * * * *  L
```

```
# !/bin/sh
# This script starts the AX.25 listen utility
#
listen -8 -acrt
```

# A Practical Approach to Implementing H.F Digital Voice in the Amateur Service

Charles Brain G4GUO
7 Elverlands Close
Ferring
West Sussex
BN12 5PL
ENGLAND
email chbrain@dircon.co.uk

## Abstract

This paper describes a practical approach to building a working digital voice system suitable for NVIS operation on the H.F Amateur bands.

## Introduction

This whole project began due to a comment a friend Andy G4JNT made over the telephone he said that it would be fun to transmit real time digital speech on the amateur bands. Now there was a challenge! As he was located some 70 km away over a fairly obstructed path it would have to be on H.F, even more of a challenge!

### Choosing the Vocoder

A number of candidate systems were studied. The Vocoder had to operate at a low data rate, be inexpensive, be standalone and relatively available. The systems looked at were, LPC-10e, MELP, AMBE and various CELP systems.

I experimented with LPC-10e and even managed to implement a version of it on a Motorola 56002EVM, the speech was understandable but I never did manage to get it to track the pitch correctly. Having listened to a commercial implementation of LPC-10e I decided that it did not have acceptable speech quality anyway.

I then went on to find an implementation of MELP (the new DOD standard) on the Internet, I managed to get the code to compile and added some Win95 sound handling routines. The speech quality was much better, but it consumed about 90% of the CPU resources on my P133 machine. Also after contacting the patent holders I was told that they were not at all happy with what I was doing.

I then looked at CELP based systems, these require large codebooks and clever search algorithms, something I thought was beyond the ROM capability of the Motorola EVM and my programming skills!

So finally I settled on the AMBE vocoder chip manufactured by DVS INC, this chip is relatively cheap, has very good sound quality, is scaleable between 2400 bps and 9600 bps and the manufacturer would sell me some!

## Choosing the Modem

After a literature search I came to the conclusion that the modem would have to use parallel tone technology. This was because it was easy to implement, was well proven, would run on my EVM and was more suitable for digital voice transmission than serial tone modems. Serial tone modems tend to produce long bursts of errors when the equaliser fails rather than the more random errors produced by a parallel tone modem. Speech is unlike computer data, in that the occasional error does not significantly affect the intelligibility.

## Designing the modem

Amateur radio equipment has very poor filtering compared to military equipment. The filters tend to be quite narrow and have poor group delay characteristics. This means the modem has to use a narrower bandwidth than the equivalent military one would. This ruled out the MIL-STD 188-110A 39 tone modem.

In the end I decided on a 36 tone modem, with a baudrate to match the 20ms frame length of the AMBE vocoder chip. This provided a raw data rate of 3600 bps and enough time for a 4ms guard period. The guard period was required to give the modem multipath tolerance. The data was modulated using DQPSK which meant that each tone carried 2 bits of data during each baud interval. Unlike military modems my modem has no Doppler correction tone and no slow sync on data facility. So far both of these facilities have proven unnecessary. The modem remains in lock for long periods of time (well beyond my ability to carry on a monologue) .

I then did some MATLAB computer simulations that showed that the modems had to be within 5Hz of the correct frequency to work properly.

To achieve initial timing and frequency offset correction the modem used three BPSK modulated preamble tones. It differentially decodes them using a delay of one baud interval it then integrates the received symbol over that time, from this it deduces the timing epoch. Then by looking at the energy in the FFT bins either side of the preamble tones it calculates the frequency error and make a correction by translating the received signal in frequency using a complex mixer. The reason for three tones is to provide some frequency diversity as on air testing showed a single tone can get lost during deep fades.

Each symbol consists of 160 samples with a sample rate of 8ks/s. The 36 tones were created by using a 128 point complex FFT, the guard period is added by taking the last 32 samples from the output of the FFT and adding them to the beginning of the FFT samples to form a total of 160 samples. These 32 samples form the 4ms guard period. The data is differentially coded and mapped to the output phases using Gray coding before transmission.

After the preamble has been sent the modem sends a reference vector, i.e. transmits a known phase on each of the 36 tones, it follows this by a sync sequence. When the receiving modem detects the sync sequence it ceases hunting for the preamble and starts passing (hopefully) valid data to the vocoder board.

When the operator releases the PTT the modem detects the loss of voice data and transmits an EOM (End Of Message) sequence embedded in the data stream, this message is in fact the SOM (Start of Message) sequence inverted. Transmit / receive control of the modem is triggered by the presence / absence of data from the vocoder, there is at present no protocol between them.

One problem with parallel tone modems is that they tend to have very high peak to mean ratios. To combat this my modem used different initial phases on each of the tones and also applied clipping and filtering of the output signal. This allowed the transmitter to be driven quite heavily before errors began to start to appear in the received signal. The simplest way to set the audio level up was to increase the drive level until ALC action occurs then back it off.

The modem is capable of full duplex operation, it does not require a feedback channel so can be used in broadcast operation i.e. one sender and many listeners.

The modem also incorporates a CW-ID feature to comply with U.K regulations, so the old meets the new. The CW callsign has to be hand coded into the DSP S/W but can be switched off. It is not sent at the end of each over but after a programmable period.

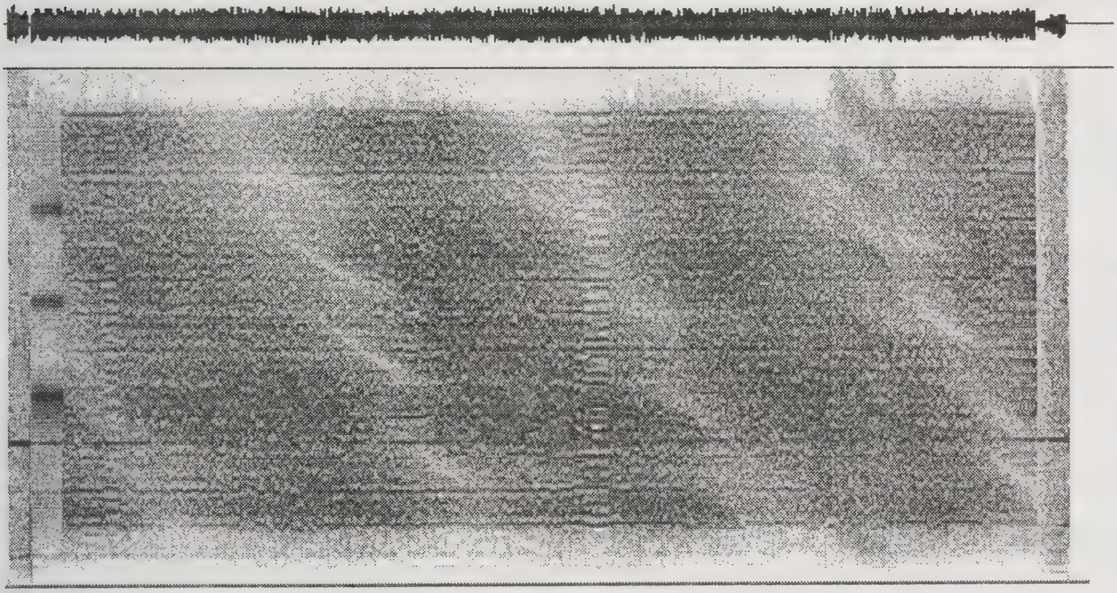


Fig 1 Off air spectrogram of M36 modem

Fig 1 shows a compressed spectrogram of an off air transmission. The three preamble tones can be clearly seen along with the selective fading (diagonal stripes), as can the carrier of an AM broadcast station in the background and a burst of interference at the end. The distinct vertical stripes were in fact pauses in the speech.

**The use of FEC**

The modem has no inherent FEC embedded in it, instead it uses the FEC in the AMBE vocoder chip itself. The vocoder tailors the FEC to match the significance of the bits in the data stream, so it can

probably do a much better job than I can. However it was a shame to not be able to use the soft decision information generated by the modem.

The AMBE chip uses both Golay and Hamming codes for correction and detection. It follows the normal convention during periods of errors, trying to guess what was sent by looking at previous frames then eventually giving up. The format used is 2400 bps speech and 1200 bps FEC.

The first tests were done without the FEC enabled, (whoops) the system worked quite well but occasionally very loud screeches came out of it, however after the FEC was enabled fewer strange noises came from the system. When the modem was initially tested without the FEC, 1/3 of the tones were in fact transmitting no data whatsoever and were therefore wasting energy.

**The use of Interleaving**

Some experiments were done using an interleaver but were abandoned because the interleaver adds a large delay to the voice and during deep fades or periods of interference spreads the errors over multiple vocoder frames and so prolongs the dropout .

**Development of the Vocoder PCB**

The vocoder board consists of a Motorola MC14LC5480P codec using μ Law coding, an AMBE chip, a PIC17C44JW microcontroller, some 74HC series glue logic and a RS232 interface. the AMBE is a 100 pin SMT chip, which had to hand solder onto the board, it was when I got up to board number 5 that I began to get tired of doing it!

For the PCB I used the services of ExpressPCB in the US. This in hindsight was a mistake as their free PCB software is not compatible with anyone else, so it pretty much locked me into using them once I had started. Their service is very good however, I emailed the files on Monday and had the boards back in the U.K by Thursday. I also sourced most of the components from Digikey in the U.S as well, as it worked out cheaper than buying them in the U.K, especially the microcontrollers.

I used the 17C44 PIC microcontroller, for a number of reasons, firstly so I could use one crystal to drive both the AMBE and the PIC ( the AMBE requires a 27-30 Mhz clock). Secondly the 17C44 PIC has enough ROM available to allow quite complex code to be added at a later date. In fact I have since done a version of my software that can encrypt the speech using triple DES encryption in real-time. Finally because I already had the development tools available.

The boards cost me about $150 each to make and I have so far made 5.

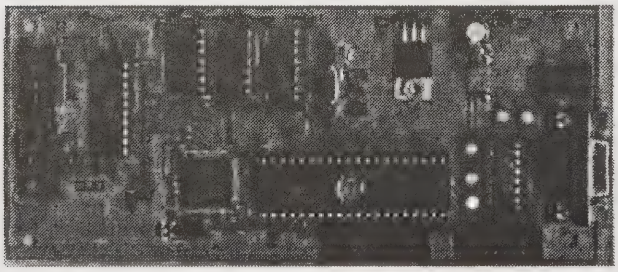


Fig 2 Prototype Vocoder board

**On air testing**

The system has been tested over a 70km path using frequencies in the 40m band. Andy and I made our first successful contact at the first attempt on the 27th of March 1999. This is not a weak signal mode and requires about 25 db S/N to function. However when working it makes H.F sound like a telephone conversation. There is no background noise, total silence, except for the comfort noise inserted during gaps in the speech by the vocoder itself. The system can tolerate strong CW interference and also the multipath induced selective fading found on H.F. SSB interference is more troublesome as it affects more than one of the tones. If RTTY/CW interference gets too bad it is even possible to switch a DSP notch filter in circuit, there is enough power in the FEC to cope with the missing tones, however the notch filter must be switched out during the preamble phase.

The weakest part of the modem is the preamble phase, to help solve this I added the ability to save the frequency offset correction and timing epoch after each successful preamble synchronisation. If for some reason the receiving modem misses the start of the transmission it is then possible to press a button on the front panel and revert to the last set of sync information. In a one to one QSO this works most times.

Another change that was made to the modem was to allow the different tones to be given different amplitudes to compensate for the amplitude response of the transceiver. The group delay in the transceiver will however reduce the modems tolerance to multipath.

With the new generation of IF DSP radios this will not a problem as their filter characteristics are much more suited to this kind of operation.

As well as H.F testing I have also used in on 2m both on SSB and FM, and there is no reason it would not work via a repeater as there is no ARQ (but I have not tried it).

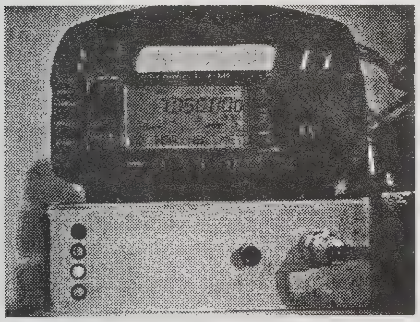


Fig 3 Current Digital Voice Station

## Conclusion

It is now possible for the home constructor to build for about $300, a portable, working digital voice system for H.F, with near toll quality audio. This system can equally be used to experiment with digital speech using different DSP modems on different frequencies.

For further information and a full technical description plus some sound files surf along to my website http://www.chbrain.dircon.co.uk/dvhf.html

# XML and APRS

Steve Dimse K4HG
189 Le Grand Lane
Cudjoe Key, FL 33042
k4hg@tapr.org

## Abstract

Recently, a number of pundits have been calling XML (eXtensible Markup Language) the "Next Big Thing" on the Web. New XML aware applications are being released all the time. Over the few years, more and more information will be made available in XML. The ability to easily machine-interpret this information will be a watershed event in the development of the Internet. APRS will not be left out! In this paper I detail the first efforts in making APRS data available in XML.

## Introduction

While some have called XML a more advanced form of HTML (HyperText Markup Language), this shortchanges XML, and fails to describe the true relationship between the two. HTML grew out of an important standard known as SGML (Standardized General Markup Language). This is a DoD standard that is used to maintain vendor-independent structured documentation. It is very powerful, very complex, and very difficult to implement. A very limited and well defined subset of SGML was initially selected for inclusion in HTML, to specifically tailor it to the needs of hypertext documents served over the Internet. Over time HTML has evolved to meet the specific needs of web users.

A serious limitation of HTML is that it only describes the way a document looks. Formatting instructions are enclosed between the characters '<' and '>'. The tag `<h1>` defines a top-level header, and causes the following text to be printed in large, bold letters. However, HTML provides no way to identify what that information represents…it might be a rock group, the title of a scientific paper, or a disease. The human reader is able to identify the meaning from the context, but this is a very difficult process for a computer to duplicate.

XML addresses this shortcoming of HTML. While sharing a common derivation from SGML, XML allows the user to define arbitrary structures since, as the name suggests, it is user extensible. Rather than formatting, the focus in XML is on the structure and meaning of the underlying data. The HTML phrase `<h1>Counting Crows</h1>` might become `<h1><rock_group>Counting Crows</rock_group></h1> in XML`. Now, for example, a Web indexing engine would be able to tell that this page would be of no interest to a corn farmer concerned about rising pilferage by birds!

## Data Tags for APRS

The most essential task is creating a standard set of data tags. There are many efforts underway to standardize the tags for specific applications. Before beginning this effort for APRS, I searched the Internet for an existing standard, but I was unable to find any XML specifications pertaining to position or mapping information. Therefore I decided to start from scratch. Part of the design philosophy of the XML standard is bandwidth is cheap. There are no cryptic tags in the interest of saving a few bytes. The result of this is that the data is very verbose, but very readable.

### Sample APRS data converted to XML

```
<station>
      <callsign>K4HG</callsign>
      <position>
            <timeReported>1999-01-16T22:32:06Z</timeReported>
            <timeReceived>1999-01-16T22:32:11Z</timeReceived>
            <positType>Mic-E</positType>
            <icon>6,43</icon>
            <lat>24.5434</lat>
            <lon>-81.5432</lon>
      </position>
      <weatherReport>
            <timeReported>1999-01-16T22:32:06Z</timeReported>
            <timeReceived>1999-01-16T22:32:11Z</timeReceived>
            <windSpeed>21</windSpeed>
            <windDirection>326</windDirection>
            <humidity>86</humidity>
            <barometer>1004.3</barometer>
            <rain1Hour>0.02</rain1Hour>
            <rain24Hour>0.25</rain24Hour>
            <rainMidnight>0.14</rainMidnight>
            <temperature>86</temperature>
      </weatherReport>
</station>
```

As you can see, this format takes considerably more storage than the native APRS formats. However, instead of the 6 different position formats defined in APRS, there is a single format that a program must parse.

## Uses of XML and APRS

Perhaps the most successful portion of my Internet/APRS software is the map.aprs.net server. This allows the display of street level maps for any station whose position is known to the APRServe, using any ordinary browser. There is a simple web server embedded directly into the software. The user requests a callsign, the server checks its internal database, and dynamically generates an HTML page, including calls to the MapBlast web server to generate the maps. Unfortunately, under this system there can only be a single page served to all users. To modify the page, I must change the APRServe software

and reboot the server. There is no way for the user to customize his web page. With XML, it will be easy for people to develop their own web page, using an HTML feature called a template, which will parse the XML data and display it in any fashion they choose.

Another possibility is taking the data, and importing it into other programs. For example, one could take the data for a balloon flight, import it into Excel, and generate a graph based on altitude.

## XMLserve

Every day, APRServe handles about 25 MB of data. No systematic archive is made of it. XMLserve will be a huge database of this data, and allow sophisticated queries to be made, for example give me my position every fifteen minutes for the last month. The results will be returned in XML, allowing the user to then manipulate or display the data in any fashion they wish. At this time I am still trying various database programs for the back-end.

## Present Status

There are examples of the preliminary XML data format, and an embryonic parser written in C++ available on my server at http://www.aprs.net/xml. My finalization of the specification is awaiting the completion of the APRS Working Group's protocol document. Comparing the XML tags to the protocol will ensure that no data format is missed. Also, I invite input from others interested in this subject to comment on the specification. Once it is complete, I plan to submit it to the APRS Working Group as a proposed standard.

The APRS XML suite and XMLserve will do nothing on their own…these are enabling technologies. What is needed are parsers both into and out of XML for other popular languages, like Perl, C, Visual Basic, and Java. I hope others will fill these gaps with open source programs. Also needed are examples of template HTML files for position and weather data. Finally, real-world uses for the data need to be dreamed up and implemented.

# Proposal for a Spread Spectrum Transponder Payload On the International Space Station

Matthew Ettus, N2MJI
Integrinautics Corporation
1505 Adams Drive
Menlo Park, CA 94025
matt@ettus.com

August 5, 1999

## Abstract

A satellite payload for the International Space Station is proposed, which would provide high-bandwidth, wide-area data communications capabilites for radio amateurs. Key features of the system are a simple space segment and low cost ground stations. Varying tiers of service can be provided depending on end-user equipment investment, from low-cost paging, through digital voice, video, and high-speed data communications.

## 1 Introduction

### 1.1 Vision and Goals

The design, construction, and deployment of an experimental payload for the EXPRESS Pallet on the International Space Station (ISS) is proposed. The payload, to be known as the Spread Spectrum Wideband Transponder (SSWBT), is designed with the following goals in mind:

- Low cost digital voice communications
- Digital Videoconferencing "for the masses"
- High bitrate, low-latency data transfer
- Development of Spread Spectrum technology in the amateur community
- Open space communications to the average amateur
- Room for future growth and expansion

We envision four general tiers of use for the SSWBT:

**Ultra-low bitrate ( $<$ 1 kilobits/second)** These would be handheld size stations, with simple patch antennas. These would be useful for paging, position reporting/homing (APRS), emergency distress beacons, and vehicle and property location systems. These systems could be made without receiving equipment if that functionality was not needed.

**Low bitrate (~10 kb/s)** Full duplex digital voice and data communications. With modern vocoders, 10 kb/s can provide quality better

than that typical of FM repeaters. The system will be capable of both user-to-user (QSO style) full duplex, as well as roundtable (repeater or traffic net-style) communications. These stations could be mobile-mounted, portable, or simple home stations. Small patch antennas would be used, thus avoiding the need for aiming or moving the antennas.

**Medium bitrate (~150 kb/s)** Digital Videoconferencing, web serving, and other modern internet-style applications. These will be stations which are more well equipped, and most likely, not mobile. These will require moderately sized (~1 foot) dish antennas and some mechanism for aiming them.

**High bitrate (1 - 1.5 Mb/s)** There will not be many of these stations, perhaps 6-10 per continent, placed at strategic locations so at least one is visible during any part of a satellite pass. These stations can transmit large quantities of data, typically requested by the lower bandwidth user stations. These could be internet access points, and could also broadcast (or multicast) high quality video feeds. This would be ideal for applications such as broadcasting meetings or other important amateur events. These stations may require large dishes with accurate pointing systems, and higher power amplifiers

One of the greatest features of the SSWBT concept is that while more complex and expensive systems with high power and gain will be necessary to transmit at the higher bitrates, nothing extra will be necessary to receive these transmissions. Thus, the low bandwidth systems, besides being useful for voice communications between comparable users, can be effectively used for such applications as web browsing, and file retrieval (ftp). Ten kilobits per second is plenty of bandwidth for requesting web pages, which would be served by the medium or high bandwidth systems. Highly asymmetric links are very useful for these applications.

## 1.2 Why ISS and EXPRESS Pallet

What has often held back amateurs from deploying more advanced digital communications systems has been the problem of critical density. High bandwidth often requires microwave frequencies and line of sight propagation. These are difficult to achieve in terrestrial systems unless there are enough users in a particular area. By using a satellite, these good paths can be guaranteed, while at the same time providing tremendous coverage area which would be impossible otherwise. The need to reach critical densities for deployment is avoided.

The International Space Station represents the ideal satellite carrier for the SSWBT. Because it will be placed on the ISS, the SSWBT can be quickly and inexpensively deployed, without the development of its own launch vehicle. It will serve as an ideal testbed for a possible future network of microsatellites, and local terrestrial transponders to provide complete earth coverage. Since the satellite will be accessible worldwide, technology and development can be shared, improving the economies of scale, and making it more likely that the system will catch on in significant numbers.

The EXPRESS Pallet is ideal for this type of experimental payload. The SSWBT will be small and light, due to its tiny patch antennas and very simple electronics. It will consume little power, probably less than 100W, due to the relatively low and nearly

circular orbit of the ISS. It will require no interaction with other systems on the ISS, and its only controls will likely be to enable or disable it. The SS transmissions of the SSWBT will not interfere with the other experiments on the Pallet or the rest of the ISS. The nadir-pointing attribute of the Pallet makes line of sight contact possible.

# 2 Technical

## 2.1 Features

- Direct Sequence Spread Spectrum (DSSS) Modulation
- 5.7 GHz Band Uplink
- 3.3 GHz Band Downlink
- 50 MHz wide signal bandwidth
- Automatic Power Control
- Scalable bitrate

## 2.2 Capabilities

This system will be able to accommodate over 500 digital voice conversations, dozens of high bitrate video conferencing sessions, and a T1-class data link, all at once. Stations within 400 miles of the point directly below the ISS will be able to access these facilities, providing a coverage area of about half a million square miles. It can provide high data rate, asymmetric data links to small mobile users, with tiny patch antennas. User systems will have low power consumption.

## 2.3 General Architecture

In order to receive and demodulate SS signals from hundreds of users at one time, hundreds of demodulators would be necessary on the ISS. Instead, the SSWBT simply amplifies and retransmits the signals which it receives. This allows the ground stations to each pick out and demodulate their own signals.

A key advantage of the SSWBT is its very simple space segment. The payload will consist of a linear transponder, and a simple "carrier" signal generator. All of the complexity will be in the ground stations. This allows for easy changes to the modulation format, and avoids the need for complex and expensive radiation-hardened DSP components.

### 2.3.1 Modulation and Coding

DSSS Modulation will be used, with binary phase-shift keying (BPSK). The manner in which spreading codes are assigned will be discussed below. Whatever the bit rate which a station is transmitting at, it will always use the same chipping rate, 25 MHz. Thus, all signals will have the same occupied bandwidth of 50 MHz, and processing gain will be inversely proportional to bit rate. Nyquist filtering will be used to keep the occupied bandwidth to less than 50 MHz. Effective radiated power will be in direct proportion to bit rate, so that energy per bit is constant for all stations.

In order to provide more reliable communications, with lower power, and higher user capacity, forward error correction (FEC) will be used extensively. The most likely candidate is Convolutional coding and Viterbi decoding. ASICs are commonly available now which are capable of high data rates with rate 1/2, and 1/3 codes and constraint lengths of 7 or 9. Other options might include combining convolutional

codes with Reed-Solomon codes, or even using turbo codes. Again, these are all issues for the ground stations, and so could be changed without touching the transponder. Different FEC systems could even be used for each of the different data rates, although that would probably not effectively reuse components. Multiple schemes could be used concurrently, allowing experimentation to coexist with normal use.

Different spreading codes correspond different "channels" of communications. Each station will have an assigned "hailing code," to which it will always be listening. When station A wishes to transmit to station B, station A will transmit using B's hailing code. In this first packet, A will send a code pair, one for a to use when talking to B, one for the reverse link. They will then use these codes for the duration of their communication. As long as the set of all codes is sufficiently large, collisions (different transmitters using the same codes at the same time) can be avoided.[1]

### 2.3.2 Automatic Power Control

Automatic power control (APC) is necessary to make this system work. Without it, stations closer to the satellite would swamp out the ones further away. APC guarantees that all signals will be received at the same strength, maximizing the number of them that can be decoded successfully.

The pilot signal will be used as the reference power level. When a station is transmitting, it must simultaneously receive and decode its own signal, as well as the pilot signal. The transmitting station must constantly adjust its power up or down to make its downlink signal equal in power[2] to the pilot signal. The actual downlink power received will vary, but the relative levels of the many signals and the pilot signal will remain the same.

### 2.3.3 Space Segment

The space module, the SSWBT itself, is a simple linear transponder, with only one addition. A simple (and small) 10 dBi, circularly polarized patch antenna receives the many uplink signals at 5.7 GHz. After being amplified and filtered, they are downconverted to IF. At IF, the signal passes through a 50 MHz wide channel filter, and an AGC amplifier. Then a pilot signal is injected, and the combined signal is then upconverted to 3.3 GHz. After it is amplified (about 25W output), it is retransmitted back to earth via another 10 dBi circularly polarized patch antenna.

The pilot signal is very crucial to the operation of the system as a whole. It allows the ground stations to have a reference power so that they are able to provide near perfect power control. It also provides a signal timing and doppler reference which the ground stations can also use to ease the problem of getting code and data synchronization.

### 2.3.4 Ground Segment

A minimal ground station, capable of transmitting digital voice, will be the typical end-user system. Such a station will use circularly polarized patch antennas, just like the satellite. It must have at least three despreading channels. One to monitor the pilot signal, one to monitor the station's own transmitted

[1] In the case of two stations transmitting on a third's hailing channel at the same time (a collision), both should detect it. Normal random backoff procedures would be used. High bandwidth, high utilization base stations should have multiple hailing channels to avoid this.

[2] Actually, energy per bit will be controlled. This will allow signals with varying bit rates on the same channel.

power and timing, and one for useful reception of signals from other stations. Since all of the signal processing associated with despreading channels will be done in digital logic in FPGAs or ASICs, adding more will not be difficult. Additional channels will be useful for receiving many datastreams at once.

For transmitting, a power output of 1 Watt (and the capability to reduce that output power) is all that is necessary for communication. Stations of this class should cost well under $1000, and could easily be made mobile. Again, while these stations will only be able to transmit at low bitrates, they can receive at the highest rates.

A high end home station, if it is to transmit at 150 kbit/s, would need to produce 15 times the effective power output. Most of this gain would to be provided by the antenna, so that commonly available integrated power amplifiers in the 3-5 W range could be used. This implies the need to point the antenna, however, and that may add to the cost of some installations. Otherwise, all hardware would be the same as the standard end user system. The receive antenna could still be the same patch as used by low end stations.

High bandwidth, regional base stations, which need to transmit in the 1 Mbps range, would have to have moderate sized dishes, and power outputs in the 10-25W range (this can be traded off against antenna gain). The receive antenna could again be the same small patch, but better results on distant passes, near the horizon, could be had with small, aimable dishes. This would give a regional base station more coverage, increasing the probability that one is always in view of the satellite.

Receive equipment on the regional base stations would be similar to the user stations, with the addition of many more despreading channels in the hardware. This would allow many more simultaneous connections and requests, allowing the station to keep its transmitter busy supplying data.

## 3 Conclusions

The SSWBT will open up a whole new world of digital communications to the amateur radio community. By taking advantage of underutilized spectrum, and advanced communications techinques, we will finally be able to interconnect the ham world with a high bitrate, integrated network. This will open up the possibility of digital videoconferencing, digital voice communications, and high speed data transfer. The ISS and the EXPRESS Pallet will make this all possible by solving the problems of line-of-sight propagation and geographic coverage.

# A Low-Cost HF Channel Simulator for Testing and Evaluating HF Digital Sytems

**Johan B. Forrer, KC7WW**

## Objective

The incentive and justification for this project was inspired by the author's desire to develop HF digital communications devices that effectively deal with the variable nature of the ionospheric propagation medium. Simulating the behavior of the ionosphere in real time allows for bench testing of HF modems and other communications devices. In the past, these so-called "HF Channel Simulators" used exotic and expensive computing hardware that was not available to the average amateur experimentor.

The simulator presented in this article is based on a low-cost floating-point DSP evaluation kit that accommodates a wide range of simulated conditions, including CCIR 520-1[1]. The simulation model is an implementation of the Watterson, Gaussian-scatter, HF ionospheric channel[2] model which is the *de facto* standard for this kind of work.

The article concludes with a summary of test results for a number of contemporary, forward error-correcting (FEC) HF digital systems tested on this HF channel simulator: PSK31, CBPSK, and MT63.

This simulator is a worthy addition to anyone's array of testing tools for developing DSP modem algorithms, routing or protocol development for HF communication systems.

## The Challenge Posed by the Variable Nature of the HF Channel

HF propagation involves several interrelated phenomena that result in a highly variable propagation medium. This variability is a challenge to anyone that needs to design and implement effective high-speed digital communications systems for HF.

The ability to quantitatively evaluate how successful engineering designs carries though to real-world implementations, often makes the difference between success and failure. Experienced, well-equipped engineers use special tools such as channel simulators to shorten development cycles. These are invaluable for example, to verify dynamic range performance, acceptable signal to noise ratio performance, as well as a number of other factors such as adjacent channel interference and frequency/timing tolerances. These are very common real-world problems. Besides the evaluation of these basic factors, protocol performance is of equal importance. This has to do with how efficient frame and character synchronization is, how effective error control works, and how successful protocol adaptation actually is.

Although some of these tests may be done by on the air tests, however, F-layer propagation conditions are almost impossible to repeat thus there is not really a chance for making comparative tests this way. What

---

[1] CCIR Recommendation 520-1. Use of High Frequency Ionospheric Channel Simulators.

[2] Watterson, C.C., J.R. Juroshek, and W.D. Bensema. 1970. Experimental confirmation of an HF channel model. IEEE Trans. Commun. Technol., vol. COM-18.pp. 792-803, Dec.1970.

really is needed is a means to create an artificial ionospheric test medium ("ionosphere in a box") that can be reproduced at will. Only then is it possible to set up norms and milestones for performance evaluation.

Computer simulation is one way to obtain quantitative results. A simulation study based on theoretical concepts can provide the basis for establishing expected performance characteristics, also serve as a guide as to requirements for hardware and software expectations. It can provide an essential justification for continuing development work without the risk.

During test and development phases, real-time testing using a HF channel simulator is essential. The key to developing an effective waveform and protocol suitable for high-speed HF digital communications, is in understanding the behavior of the ionosphere and how it will impact communications.

## Ionospheric Reflection Model

HF communication is typically characterized by multipath propagation and fading. Transmitted signals travels over several propagation modes to the receiver via single or multiple reflections from the E and F ionospheric layers. Because of different propagation times over different paths, signals arriving at the receiver may be spread in time by as much as a few milliseconds.

Ionospheric turbulence causes distortion in both signal amplitude and phase, in addition, different ionospheric layers move up or down, which leads to independent Doppler shift on each propagation mode. Ionospheric skywave HF, multipath arises from paths with different number of multiple reflections between earth and the ionosphere (multiple-hop paths) and from paths at multiple elevation angles connecting the same end points ("high" or "low" rays). Natural inhomogenuities of the ionospheric layers and polarization dependent paths because of magnetic-ionic effects also contribute to multipath.

The effect of these natural inhomogenuities in the ionosphere causes multipath spreads of 20 to 40 µs on each path or mode, and the high/low and ordinary/extraordinary rays results in a path spread of about 200 µs. For single hop links (800-2000 km), a maximum multipath spread of 100 µs is common. In this case, all paths are via the same reflection area and thus there is no significant difference in the Doppler spread on different modes. The channel is often a very slow fading channel, with time stabilities of 100 s or more, corresponding to a Doppler spread of 0.01 Hz. Multipath spread in the range of 1 to 2 ms for HF occur for short ranges (because of near vertical incidence) of under 800 km due to delayed energy arrival via repeated earth-ionosphere reflections or over long paths (2000 to 10000 km) that require two or more hops. On these long skywaves, different spread, controlled by the Doppler shift differences can easily range up to 1 to 2 fades per second.

Short-term distortion on the HF channel can therefore be described in terms of the parameters that specify the time-spread and frequency-spread characteristics, i.e., differential propagation delay between modes, and the strengths, Doppler spread on each mode.

Figure 1 shows an actual example of these different mechanisms in action. *(This illustration provided by courtesy of J.P. Martinez[3].)* Martinez experimentally recorded an event on November 9, 1994 that by saving a digitized audio tone of a remote broadcast station's carrier on a computer file. The broadcast station's carrier was located on 7.7 MHz and arrived via the ionosphere; the broadcast station being located on the island of Gibraltar and the receiver located on the South coast of England. Subsequent processing of the recorded digital data revealed frequency-domain behavior over time. For this, the results of 256-point FFTs are presented as pixel intensity values on the Y-axis, with time plotted on the X-axis.

---

[3] Martinez, J.P., G3PLX, High Blakebank Farm, Underbarrow, Kendal, Cumbria LA8 8BN, United Kingdom. The author gratefully acknowledges J.P. Martinez's permission to reproduce these experimental results.

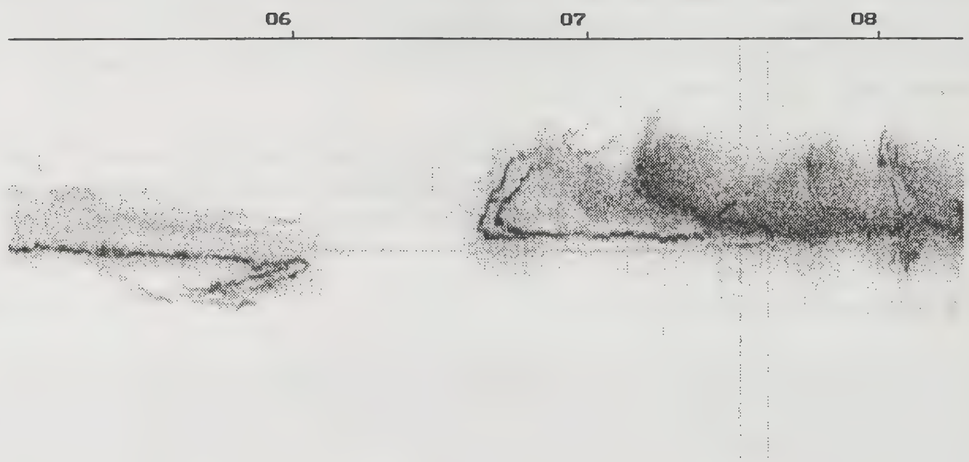



Figure 1. Martinez's Dopplergram illustrating several interesting ionospheric phenomena.

For the graph shown, each pixel point in time represents approximately 20 seconds of signal with UTC hour tic marks shown along the top. The Y-axis represents 0.025 Hz/pixel (256 pixels=6.25Hz). This representation effectively shows the history of a very slowly-changing process, with most of the finer, random events, filtered out to better illustrate the various propagation modes.

Because of the frequency in question (7.7 MHz), we are reasonably sure that the propagation mode is most likely via the F-layer. Note that at about 06:00 UTC the signal penetrates and no signal propagation path to Earth results. Just before this happens, note the high F-layer ray (the so-called, Pedersen ray) appear lower in frequency than the main (low) ray. The high ray itself appears to be split in two parts each with distinct Doppler shifts; the upper image being probably being the opto-ionic, or O-ray, and the lower image being produced by the extra-ordinary, or X-ray. The X-ray undergoes further retardation due to interaction with Earth's magnetic field. Shown is that the high and low rays of the O-trace penetrate first, followed by the X trace. This effect is distinct on this Dopplergram, but only rarely is it identifiable by ear.

If recognized, it appears as regular fading (QSB) that slows down to zero as the particular path fades out. About 06:40 UTC the F-layer comes back in again and the process is seen in reverse, X-trace appearing first and splitting into high and low, followed by the O-ray. Further more diffuse propagation paths open up a few minutes later.

## The Watterson Gaussian-Scatter HF Ionospheric Channel Model

Watterson et al, using wide-band HF emissions over a path between Bolder, CO. and Washington, DC., proposed a model for narrow band HF channel. This model forms the basis for most modern HF channel simulation work and often are used for both software and hardware channel simulation.

This model, known as the "Watterson Gaussian-scatter HF ionospheric channel model", assumes that the HF channel is non-stationary in both frequency and time, but considered over small bandwidths (<10 kHz) and sufficiently short times (<10 minutes), most channels can be considered representative by a stationary model.

The HF channel is modeled as a tapped delay line, with one tap for each resolvable mode (or path) in time. The delayed signal is modulated in amplitude, and phase, by a complex random tap-gain time-dependent function that is defined by:

$$G_i(t) = G_{ia}(t)\exp\left(j.2\pi.f_{ia}.t\right) + G_{ib}(t)\exp\left(j2\pi.f_{ib}.t\right)$$

Where a and b subscripts denote the i-th element in a time series representation for two magnetoionic path components. In this context, $G_{ia}(t)$ and $G_{ib}(t)$ represents two independent complex bivariate Gaussian ergodic random processes, each with zero mean and independent real and imaginary components with equal RMS values that produce Rayleigh fading. The exponentials provide frequency shifts $f_{ia}$ and $f_{ib}$ for the magnetoionic components in the tap-gain spectrum. Each tap gain has a spectrum $H_i(\lambda)$ that, in general, consists of the sum of two magnetoionic components, each of which is a Gaussian function of frequency, as specified by:

$$H(\lambda) = \frac{1}{\left(A_{ia}.\sqrt{2.\pi.\sigma_{ia}}\right)}.\exp\left(\frac{-\left(\lambda-\lambda_{ia}\right)}{\left(2.\sigma_{ia}^2\right)}\right) + \frac{1}{\left(A_{ib}.\sqrt{2.\pi.\sigma_{ib}}\right)}.\exp\left(\frac{-\left(\lambda-\lambda_{ib}\right)}{\left(2.\sigma_{ia}^2\right)}\right)$$

where $A_{ia}$ and $A_{ib}$ are component attenuations and the frequency spread on each component is determined by $2s_{ia}$ and $2s_{ib}$.The frequency shift on the two components are given by $\lambda_{ia}$ and $\lambda_{ib}$.

Tap-gain distributions for a two-ray model are shown in Figure 2.

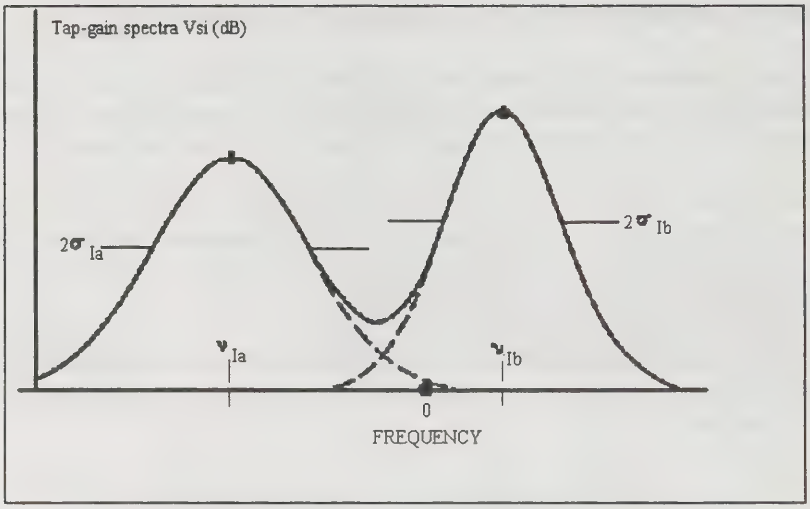



Figure 2. Tap gain distributions for a two-ray model.

## Notes:

1. The Watterson model implies the use of equal power (RMS) paths. This effectively is like a deep notch filter sweeping through the passband – at times completely obliterating parts of the signal. This often has devastating implications for some modem algorithms and some end users of this simulator has expressed their concerns as "it not being realistic for typical HF conditions." In order to reduce the depth of the null, it is possible to weigh tap gain functions such that they are never equal, however, this practice should be for in-house developments only and not for publication as such results will include unjustified bias.
2. In attempts to compare performance results of standard equipment against published materials where professional channel simulators have been used (manufactured by Harris Corp. for example,) it has been found that there appears to be some leeway in interpretation of the Watterson model and subsequent discrepancies in results. There has been investigations by researchers on this subject , however, without having access to details on proprietary implementations, these discrepancies remain unresolved.
3. Generally, published specifications or research results often tends to omit weaknesses that are readily shown by such simulators. More often than not, results obtained by this simulator tend to be interpreted as highly critical or erroneous. This is not the intention, rather should be an opportunity that should be exploited to the user's advantage.

## CCIR Recommendations for the Use of HF Ionosperic Channel Simulators.

CCIR Recommendation 520-1 gives guidelines for practical values for frequency spread and delay times between ray components:

| ***Condition*** | ***Freq. Spread (Hz)*** | ***Delay (ms)*** |
|---|---|---|
| Flat Fading | 0.2 | 0 |
| Flat Fading (extreme) | 1.0 | 0 |
| Good | 0.1 | 0.5 |
| Moderate | 0.5 | 1.0 |
| Poor | 1.0 | 2.0 |

It is proposed that these parameters be used to validate average and extreme conditions during simulation as well as during actual hardware testing.

## The Development of a Real-Time HF Channel Simulator

Discussions on developing a low-cost HF channel simulator took place on several forums; TAPR HFSIG list, specifically during 1994, 1995 TAPR Annual Meeting in St. Louis, MO., Digital Communications Conferences (DCC), 1995 Arlington TX, and 1996, Seattle, WA.

Early work involving Alexander Kurpiers, DL8AAU, Darmstadt Germany, produced code for a TI 320C26-based DSP implementation. The author ported this for use on the TAPR DSP93 and demonstrated its use at the 1996, DCC meeting in Seattle, WA. This model has seen service in several projects, however has limited performance due to memory and processor limitations.

Several others shown active interest in this project; Barry Buelow, WA0RJT, Jon Bloom, KE3Z, Eric Silbaugh, Glen Worstell, KG0T, Phil Karn, KA9Q, and especially Tom McDermott, N5EG. Tom presented a paper on theoretical aspects of HF channel simulation at the 1996 DCC HFSIG meeting.

The specifics for the implementation of the Watterson Gaussian-scatter HF ionospheric channel model follows. This topic is divided into two sections: the hardware platform and software implementation.

## HF Channel Simulator Hardware

The author realized the opportunity when a new floating point DSP evaluation module (EVM) by Analog Devices[4] became available. The EZ-KIT Lite SHARC is a 40 MIPS processor that can produce 150 MFLOP performance in floating point. The SHARC DSP follows modern trends where its instruction set is optimized for use with the C programming language.

The kit was supplied with GNU-based C tools on CDROM that included the usual compiler, linker, and librarian tool chain. The ability to use a high-level language made the implementation of the Watterson-model mathematics much easier. Even time-critical code like interrupt handlers may be written in C, alternately, either in-line assembly or assembly-language modules may be developed. The EVM contains a 48kHz stereo CODEC to handle audio I/O, also a UART chip to handle serial communications with a host. The DSP contains a total of 16K 48-bit words of on-chip memory, part of which is available for user code. The amount of on-chip user memory is adequate for implementing the Watterson-model simulator.

## HF Channel Simulator Software

A paper by Ehrman et al.[5] provided basic implementation ideas that was used in this project. Several parallel tasks can be distinguished:

1) Transform and process the baseband input signal such that its phase and amplitude properties can be manipulated in real time,
2) Simulate, independantly, in real time, a pre-defined HF propagation condition,
3) Apply simulated distortion to the processed input signal, and,
4) Apply noise pertubations.

---

[4] Super Harvard Architecture Computer (SHARC) EZ-KIT Lite. Part number: ADDS-2106X-EZLITE. Available from Analog Devices distributors. Street price $179.
http://products.analog.com/products/info.asp?product=21000-HARDWARE

[5] Ehrman, L., L.B. Yates, J.F. Eschile, and J.M. Kates (1982.)
Realtime Software Simulation of the HF Radio Channel. IEEE Trans. on Communications, August 1992, page. 1809.

Figure 3 shows the interaction between a number of parallel tasks. Input is applied at the top left and output produced at the bottom right of the figure.

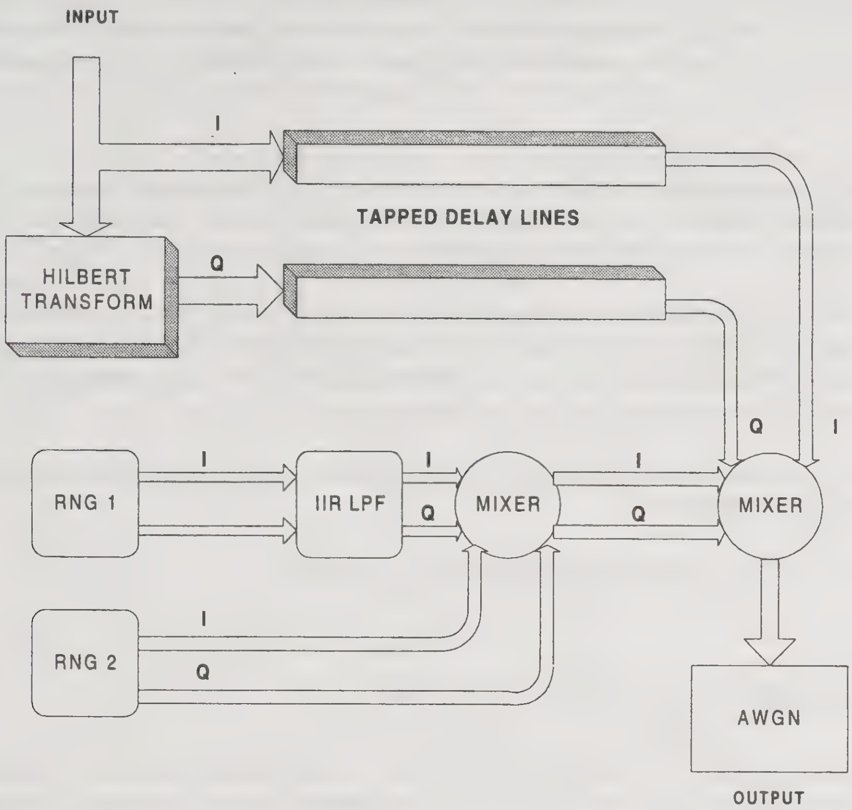



**Figure 2. Simulator Process Flow.**

The Watterson model only deals with the effects of the ionosphere and the distortion that it introduces -- it does not attempt to simulate HF noise pertubations. CCIR 520-1 also does not specify any kind of noise source, however alludes to including a noise source in simulation.

These processing steps are now analyzed in further detail:

## *Input Signal Processing*

The input signal is a real signal. Fading and Doppler effects will be introduced to this signal by a process of signal mixers. These mixers, however, are *complex* devices requiring in-phase (I) and quadrature (Q) components, thus requiring that the input signal be an analytic signal. This conversion of the input signal is achieved by using a Hilbert transform.

To simulate multiple rays passing through the ionosphere, dual tapped delay lines are used; one for the I component, another for the Q component. The analytic input signal is then extracted from the appropriate points in the delay lines -- the position in the delay line is a function of the input sample rate (typically 9600 SPS) and the required path delay (varies between approximately 0.1 mS to 10 mS, or 1 to 96 delay line taps).

### *Computing Channel Effects: Doppler Shift and Fading*

Watterson et al. showed that the desired fading and Doppler shift can be introduced by the product of two Gaussian functions, i.e., a Rayleigh distribution. Since this multiplication process of the two Gaussian functions are commutative, it does not matter what gets generated first; the fading function or the Doppler shift.

Assuming the fading function, that gets produced from a random number generator with Gaussian distribution output. This stream of numbers are then passed through an infinite impulse response filter (IIR) designed for appropriate bandwidth, i.e., that determines the fading bandwidth. Actually it controls the statistical spread for this Gaussian function, like that shown in Figure 2.

Doppler shift is produced on the fading function using a similar method, except that no filter is used. After performing the *complex* mixing of the fading and Doppler functions, the resultant signal now has a Rayleigh distribution. That is the desired tap-gain function, or modulation function to be applied to the delayed analytic input signal. The final outcome is to take only the real part of this last mixing step.

As an option, noise perturbations with the correct amplitude are then added to set the noise background for the desired signal to noise (SNR) level.

The computation of the noise background requires further consideration.

### *Computing Channel Noise Effects and SNR*

Gaussian noise models are commonly used in VHF, UHF, and microwave work, however, HF noise behavior is more complex and sometimes described in terms of Markov models, rather than stochastic models, in the literature. For purposes of this paper, only Gaussian noise is considered – this simplifies matters, however, does not accurately represent HF channel noise.

The exact channel measurements that typically are used for comparing systems should be carefully considered. Classical reference books use bandwidth-normalized SNR measurements. This reflects a unit of "bits per second per Watt per Herz" instead of a simple signal to noise ratio values. When dealing with real-world communications systems, however, this kind of measurement is difficult as power measurements need to accurately known at exact bit timings in order to compute the actual energy per bit. Coding schemes and ARQ protocol issues further complicate this measurement. It often is more convenient to determine throughput rate instead, but there would be difficulty to relate this to Eb/No as used in reference materials.

In this regard, Leeland's[6] discussion on methods to determine bit-error rates (BER) is of interest. It is suggested that BER should be this the basis for evaluating modem performance – if it doesn't meet BER specifications, it doesn't work as expected. That may imply that defensive actions like dynamic protocol adaptation and/or tracking algorithms are failing to assess channel properties correctly. BER also allows one to compose the classic "waterfall" BER vs. SNR curves. These sets of curves allows one to check measured performance against theoretical (expected) performance, but also to compare your work against other published work.

Allowing remote requests through the modem's host control port can retrieve performance measurements can assist algorithms doing a better job; Raw BER, corrected BER, and Eb/No comprise the standard suite of measurements. Raw BER is the actual count of erroneous data bits detected and corrected by the

[6] Leeland, Steven. Digital Signal Processing in Satellite Modem Design. Communication Systems Design, June 1998.

decoder. Corrected BER is the estimated BER after the decoder has reconstructed the original data stream. Eb/No is, of course, the signal-to-noise (SNR) ratio.

Historically, raw BER has been measured within the decoder circuitry by counting the number of detected/corrected bits over fixed time durations. The error count register formed the address to PROM based LUTs to supply the actual raw BER in $x.y*10^{-z}$ format to the control processor. When the BER gets as high as the $10^{-3}$ region, or some other arbitrary value, the decoder is usually ready to give up the ghost and declare loss of lock.

Through a set of arcane heuristic algorithms, the same lookup PROM generates the estimated corrected BER and Eb/No. Due to resolution, there is an upper limit to how well a BER can be measured with this technique. When these limits are exceeded, the results are reported as less than $1E^{-4}$ for raw BER, less than $1E^{-9}$ for corrected BER, and greater than 9.9 dB for Eb/No.

Modern modems use calculated Eb/No methods for BER estimation. The Eb/No is calculated from the measured SNR using symbol data. The SNR is computed from the mean, Mx, and the variance, Sx, of the data as follows:

$$S/N = (\mathrm{Mx})^2 / (\mathrm{Sx})^2$$

where

$$Mx = \sum (Xi) / N,$$

and

$$(Sx)^2 = \sum ((Xi - Mx)^2 / (N-1))$$

For BPSK and QPSK, Xi is the absolute value of the I-Channel data. For 8-PSK, Xi is *Sqrt*(I*I+Q*Q). The sample size, N, should be as large as is feasible. In order to maintain a report rate of 1 sec at say 200 symbol rate, the sample size is constrained to 200.

For some modem implementation, there are three problems with this scenario. I and Q data are digitized on both the falling and rising edge of the symbol clock. Only one edge will be correct after the Costas loops are locked. The problem is that the digital Costas loop circuitry knows which edge is correct, but the DSP does not. Another problem can be gleaned from the form of the equations given. The variance equation requires knowing the mean of the entire sample set before calculating each term in the summation.

This requires storing the entire sample set in DSP memory. Internal DSP memory is insufficient for the task, and external memory is an undesirable expense in both cost and, more importantly, board real estate.

The third problem is the square root operation required for 8-PSK:
It is not trivial to find the kind of bit edges that produce high levels SNR. For example, how does the algorithm know which edge to use, falling or rising, for the I and Q data measurements? Of course, this kind of algorithms often comes at a price – it will consume additional DSP execution time resources.

A C-code snippet shown in Listing 1 shows one approach to computing SNR. It is shown that it is no longer necessary to first compute the mean of the entire sample set. Instead, the algorithm only computes the *sum* of the samples and the *sum of the samples squared* .

```
Float snr(int sum, int sum2, int samples)
{
```

```
float mx, mx2, mi2, sx2;
        mx = ((float)sum)/samples;
        mx2=mx*mx;
        mi2=((float)sum2)/samples;
        sx2 = mi2 - mx2;
        return mx2/sx2;
}
```

**Listing 1. Implementation of SNR calculation in C.**

The 8-PSK samples require a square root operation from the specifications given in Listing 1. This is a very undesirable operation for the DSP to perform on each sample in the set of data. It consumes valuable DSP execution time resources. It requires finding and testing a square root routine.

To resolve this dilemma, I and Q data are first absolute valued. This essentially folds the eight points of the 8PSK constellation into two points in the first quadrant. To fold these two points into one, the I and Q data are compared. The larger value is used as the sample value. This is the same as comparing I and Q. If Q is greater than I, then swap I and Q. Finally, use the I value as the sample, the same way as in BPSK or QPSK.

The Eb/No is calculated from the SNR as follows:

$$Eb/No = 10*\text{Log}((1/2)(S/N)(1/c)(1/p)) - M$$

where $c$ is the code rate, $p$ is the symbol packing rate, and $M$ is the modem loss (nominally 0.5 dB).

The packing rate is 1 for BPSK, 3 for 8-PSK, and, normally, 2 for QPSK. However, because we are using only I data, $p$ is also 1 for QPSK.

If Reed Solomon decoding is installed and enabled, then:

$$Eb/No = Eb/No + 10*\text{Log}\ (N/K)$$

where N and K are the Reed Solomon encoding factors.

Finally,

$$Eb/No = \text{Fudge}\ (Eb/No)$$

where "Fudge" is a function that accounts for differences between theoretical versus real-world modem situations.

## Test Results

Simulator tests were performed on three FEC communications modes: PSK31, CBPSK, and MT63 as examples. In this example, the test condition used was CCIR POOR, which comprises the use of two equal-power rays with 2ms differential path delay, 1 Hz Doppler frequency spread. The SNR level was set at -10dB SNR. This represents a 3kHz bandwidth AWGN channel. This test condition represents marginal HF conditions, that probably are close or at the practical limit for reliable HF communications.

Results are shown in Appendix 1.

## Acknowledgements

This work was made possible by generous contributions made by participants of the TAPR HFSIG list and discussions at various DCC meetings. Not only did these forums stimulate the development of this HF channel simulator, also new HF digital communications modes like PSK31 and MT63.

The author gratefully acknowledges the contribution of TAPR in this respect and wish to thank those that participated in the multitude of interesting and educational postings on the HFSIG list.

The contributions of Peter Marinez's, G3PLX, ionospheric "Dopplergrams" as well as work on PSK31 is gratefully acknowledged.

A special word of appreciation to Pawel Jalocha, SP9VRC who brought us SLOWBPSK, the granddaddy of PSK31 and MT63.

Free demonstration simulator code is available for downloading[7] from the author's world-wide web site.

[7] http://www.peak.org/~forrerj

## Appendix 1

Simulator tests results performed using PSK31, CBPSK, and MT63 under CCIR POOR conditions (two equal-power rays with 2ms differential path delay, 1 Hz Doppler frequency spread) at -10dB SNR, 3kHz Bandwidth AWGN.

The contents of the test message is the "TUNER program" as shown. The results after passing the test message through the simulated channel using the selected HF communications mode are shown.

*Notes:*
1. Due to decoding errors, some unprintable control characters were encountered that caused the word processor to make substitutions, more often than not, line feed characters.
2. The last test for the 2kHz bandwidth MT63 used –5dB SNR.

## The "test" message:

<u>The TUNER program - TUNER.COM</u>

1. This is a tuning aid to help get a received tone exactly on 800.0 Hz.
It should accept COM2, COM3, COM4 command line parameters (default is COM1)
and report CLIPPING (audio signal too strong for the sigma-delta circuit).

2. Unfortunately it takes too many computing cycles to incorporate this
in COHERENT, so run TUNER first if necessary, using an 800 Hz sinewave
with no modulation on it (a steady carrier in other words).
It may be slightly useful on a carrier that is phase-modulated, but
the indicator will jump around trying to follow the modulation, and in
any event the useful frequency range would be limited.

3. The idea is to get the little yellow line centered between the 2 green
lines, and staying within the green lines at all times. The nominal
frequency is 800.0 Hz.

4. The range of this tuning indicator is 800 Hz plus or minus 20 Hz.
If your signal is not ALREADY tuned to within better than 20 Hz, this
indicator will be useless and quite likely confusing as hell!

5. There will be some rejection of other signals outside this range, but
if the signal you want is weak and the interfering signals are strong there
will no doubt be problems.

6. If you can hear the tone, there is no substitute for zero-beating it
with a good crystal-derived 800 Hz sinewave sidetone.

7. TUNERC.COM is for anyone who still uses CGA graphics - I slowed down
the update rate to accommodate sluggish LCD displays.

VE2IQ - November '95.

## Simulator Results; PSK31 with Varicode

The UNER0 on ramD TðER.ROO
-r- tiDi---iDe ---------ul----

1. Tt=s is a tuning ai t tfheð ge a repotvedi/e l tc&a ð 800.0 Hz.
It trould a oc?t Cr M2o COOððr MM cemmand line farao etera
default is ttO01)
nnd report CLIPðð Maudl signð-oo stroog for ðe oigma-deiit circðe0ð
2. Unftrtunately 7 takes too many coeeputing cycleðo
oraorate this
tn CO ERENet, so run AUl ERKirst f ne eessarð using8aeD Hz sinewað
with vtmosul ti/ on it(a stead t carri=n otrer wordo).
At o aybe slightly uð ul on a carier that s pha8-ðdulat d, but
the iodiahto ai oa juee]arount trying-f lrow the modllation, and in
any even th u eeul frequ ncy r a ae woðdbe liðte .

ð Tme ic a is to get t& littliyellow line cen Ved betweefta2tireen
ðnes, and staying wtthiihhe yreei lines at alðimesEii he nominal
frequenc io 80gbte ð.
14. The ron ne of this tuningyodicator is 800 rðplus oa ðnur 2ðHz. ebf yoeer signae es ntt yieREðY tu ld o wVain betoer taan $g Heret this
indicncr hill be us$ess Ld puite likelm honfus_gas helle

5. Theri ailLb
eome re:eðioa of othetðgnals tutsi T this raegð but
im hlsia nal you wtnhis geak and the interferinte signalalrst tng therqwill no doubt be problems.

a. _f yol can hea e the eone, thite iDno substTðe foi >ero tbeatie?wð a good crð al-der ved 800 r z si(vave si tetooer

C. TUPE eC.COM ð dð aoaone i6o stilTZ es c( graphics u I ðoe e6down
hup tat . te to adcommo rte sluggish PCD disðays.

VE2IQ - Gog]'r '9$.

## Simulator Results; C-BPSK, ET-2

Thn TUNER program -4TUNER.COM

---------------E-------------

1. Thiy is a tuning aid to help get a
Ered+i8nd tone exactly on 800.0 Hz.
It should accept COM2, COM3, COM4 c
ommand line paramKtebs (de@aul' is COM1)
and report CLIPPING (audfo sig
nal too strong for the sigmp-delta circuit).t

. Unfortunately it tak
es tooEmant computing cycles to incorporate thsm
n COHERENT, to rcn TUN
ER first if necessary, using an 800 Hz sinewave
citp no modulation on i
t (a steady carrie1 in other words).
It may be slightly useful on a car
rier that
phase-modulated, but
t_e indicator will jump_around tryidg
4to follow the modulation, Gnd in
any event the usedul frequency range
would be limited.
e
3t The idea is to ge0 the little yel.ow line cente
red between the 2%green
lines,__nd sta-ing within the green lines;at al
l times. The nominal
frequency is 800.0 Hz.

4. The range of this t
uning indicator is 80f Hz plus or minos 23 Hz.
If your signal is not AL
READY tmnOd to within better than 20 Hz, this
indicator will be uaeless
and quite likely confusing as hell!
}
5. There will be some rejection
OofEoeher signals outside this range, but
if thesi9nal you4want is weak
and the int_rfering signals are strong there
gwill no doube be probleds
.
6. If yod can hear the tone, there is no s#bstitute for zero-beati
ng it
with a geod+crystal-derived 800BHc:sinewave sidetone.

<. mUHE
RC.CE\q5;T7wS_zunwg1sW_2kT=aRh
_es CGA graphics - I slowed Town
the up

datb rate to aTco1modate sluggish LC4 displays.

1VE2IQ - Novemeer '95.

## Simulator Results; MT63 - 2kHz, double interleave factor.

The TUNER program - TUNER.COM

-------------------

--------

1. This is a tuning aId to help geT a received tOne exactly on 800.0 HZ
It should accept COM2, COM3, COM4 command line parameters (defaUlt is CoM1
and report CLIPPInG (aUdio signal too stRong for the sigma-delta circuit).

i

AG5q 2. Unfortunately it takes too many computing cycles to incorPoratE this
/kin COHERENT, so run TUNER first if necessary, usiNg an 800 Hz sinewave
with no modulatIon on it (a steady carrIer in oTher words).

I _ It may be slightly useful oN a carrierthat is phase-modulated,but
the indicator will jump around trying to follow the modulation, and in
* bany event the useful frequency range would be limiteD.

= ?jq3. The idea is to get the little yellXw line centered between the2GReen
lines, and staying within the green lines at all times. Thenominal
dfrequency is 800.0 Hz-

* x __44. The range of this tuning indicator is 800 Hz plus or minus 20 Hz.
9x m If your signAl is not ALREADY tuned to within better than 20 Hz, thIs
x - m~lindicator will be useless aNd quite likely confusing as hell!

iQe 5. There will be some rejection oFOtHer signAls outside this range, but
if tHe siGnal you wAnT is weak and the interfering signals are strong there
LB ut5Jll l_ ll no doubt be problems.

6. If you can Hear the tone, there is No substitute for zero-beatingit
u with a goOd crystal-derived 8_0 Hz sInewave sidetone.-

-

6 d2 7. tUNERC.cOM is for anyOnEwho still uses CGA
* the update rate to accommodate slUggisH LCD disPlaYS.
~ iP __VE2IQ - November '95.

**Simulator Results; MT63 - 2kHz, double interleave factor**
**(Test at -5dB SNR, 3kHz Bandwidth AWGN.)**

The TUNER program - TUNER.COM
-----------------------------

1. This is a tuning aid to help get a received tone exactly on 800.0 Hz. It should accept COM2, COM3, COM4 command line parameters (default is COM1) and report CLIPPING (audio signal too strong for the sigma-delta circuit).

2. Unfortunately it takes too many computing cycles to incorporate this in COHERENT, so run TUNER first if necessary, using an 800 Hz sinewave with no modulation on it (a steady carrier in other words). It may be slightly useful on a carrier that is phase-modulated, but the indicator will jump around trying to follow the modulation, and in any event the useful frequency range would be limited.

3. The idea is to get the little yellow line centered between the 2 green lines, and staying within the green lines at all times. The nominal frequency is 800.0 Hz.

4. The range of this tuning indicator is 800 Hz plus or minus 20 Hz. If your signal is not ALREADY tuned to within better than 20 Hz, this indicator will be useless and quite likely confusing as hell!

5. There will be some rejection of other signals outside this range, but if the signal you want is weak and the interfering signals are strong there will no doubt be problems.

6. If you can hear the tone, there is no substitute for zero-beating it with a good crystal-derived 800 Hz sinewave sidetone.

7. TUNERC.COM is for anyone who still uses CGA graphics - I slowed down the update rate to accommodate sluggish LCD displays.

VE2IQ - November '95.

# A Perspective on Open Source, Xastir, Amateur Radio and Linux

(KC0DGE) Frank Giannandrea, fgiannan@eazy.net

Linux and the Open Source way of thinking have recently been in hot debate throughout the world. Open fighting on this subject has brought out merits on either side of this discussion. Each point of view is applicable in different ways. This is my attempt to discuss how Open Source may be used to benefit Amateur Radio, and how I have used these ideas in my own project.

## What is Open Source

Open Source can be briefly described as a freely open development of a piece of software or any product containing computer code. That is, one where the author or producer gives out his/her code and/or schematics for those interested in using that product or software. The idea is to give anyone interested all the information that was used to produce that product or software. In this case, Open Source can refer to knowledge of the inner workings of a product.

Differences in the definition of Open Source are where the battles come in to play. On one end of the spectrum, some feel that Open Source also means free in all terms: freely available source, free to down load, free to use in any way. Some feel that this is the only way software should be distributed, and that large companies, such as Microsoft, are surely evil for asking money for their software, and even more so because they don't give the software's code out for all to see. On the other end of the spectrum are those who feel that there should be limits on what users have rights to, and want to see income for their time and hard work.

Over time, I think that people will discover that there is a way to incorporate all of these views, as each has its own applications. Commercial ventures can limit their licenses to benefit from Open Source without freely giving their product to users; hobbyists can work their particular ideals into a license to accomplish their goals. Open Source should be considered the means of getting people involved with a project and allowing anyone to help the project grow and change. This can be accomplished according to the preferences of each project's creators.

Allowing many different people the opportunity to add their knowledge to your project can be a great resource. Having them add code, fix problems, or work on documentation can save you time and add new ideas and perspectives. With Open Source, contributors to the project tend to work on their own areas of expertise or interest, which allows you to concentrate on what you want to do with the project and the things that you enjoy. It also forces you into the role of project manager or team leader, where you decide which properties to add and what new directions the project will take.

In every flavor of an Open Source development, no matter what perspective, source code or major parts of it are freely open. It is merely the end user license that has differences in restrictions.

## Open Source and Linux

Linux is an operating system that has also been in hot debate. It is a well-known example of what Open Source thinking can do: Linux was thought up by one man, then grown by thousands, gaining the strength to seemingly threaten even the multi-billion dollar company, Microsoft. Being able to make that claim is a feat on the same scale as that of those flaky startups, Microsoft and Apple computer, who revolutionized the computer industry in their time. Just as Apple and Microsoft created then, Open Source ideals have created a new way of thinking that may bring on a revolution of new possibilities.

When Linux was first developed, it was just part of a solution. It was the core part of an operating system, but it lacked many of the things that people would need to use it. Even before Linux, GNU was formed, with a mission to build a Unix-like operating system that was completely free. This was a strange idea at the time, but suddenly GNU had most of the pieces that Linux was missing, such as services, compilers and support-level software. In turn GNU found that Linux provided the necessary operating system. The two together form the basis of the Linux system, while hundreds and thousands of other pieces make up the rest. A large group of people, spread around the world, are freely making Linux what it is today, and making it better all the time. The idea was not new, but the idea of developing a project of this type on such a large scale certainly was new!

A common interest and Open Source ideals have brought together a great resource of people with real knowledge to build upon an open-code base. In this environment their many egos and talents collide, and somehow they manage to produce something. People with new ideas place them out for all to discuss; programmers create a product; others fix the mistakes they left behind; still others make it run faster and add new features; together they inspire those that can port it to other operating systems; somewhere in between are the artists who make it look nicer and work better. All these people work together, and in the process create a standard on a global level, all in plain view.

## How can Hams Benefit

In some ways HAM radio is uniquely suited for Open Source projects. HAM is specifically aimed at the art and science of communication. Open Source is also about communication and doing collaborative work. Its ideals bring into play open standards for communications between unlike systems. Most of the Internet's ability to communicate across diverse machines is based on open software ported from one platform to another. Programs we use every day (like sendmail, pop3d, old NCSA httpd, ftpd, and more recently Apache and perl) were created in the spirit and ideals of Open Source and were distributed, almost solely, with source code. Any one who down loaded these programs could freely modify the application, fix problems or add functionality to better suit their needs. Those who couldn't add to the code benefited as well by receiving faster fixes, more stable code, and new features. These are things that small groups working on weekends or evenings can particularly benefit from. Freely available source code can bring many small groups or individuals together to form larger groups with the same interests, which would benefit almost any project.

Since Hams commonly communicate (or at least attempt to) with people all over the globe, using Open Source thinking can make projects available to many more people. Anyone can make modifications so

those programs or devices work with their native language and local equipment. If there are standards to follow, doesn't it make sense to have them available to all Hams, not just to the ones who can work in the English language? In the APRS(tm) arena alone, how many others can't use the system because it is not in their native language? And since Ham radio is about communication, doesn't it make sense to allow anyone the ability to use and add to these projects?

Recent predictions that Ham radio is getting stale, that it is not interesting enough to recruit new Hams, make this issue even more important. Open Source allows people who may not be programmers or even involved in Ham radio an opportunity to look at some technology. Those who play with the code or device may find the talent within and make a contribution, creating more interest in Ham radio. It may even make some of us a little smarter in the process. Open Source provides the benefit of spreading knowledge, allowing technology that is usually hidden to become available to anyone who cares. If one person is inspired to add to the community effort, then many more interesting projects may develop to entice people. This can only add wealth to the Ham community and to the hobby.

## Commercial Projects

Most groups opposed to Open Source projects feel that their intellectual property is at risk. This certainly may be the case in most industries. The Open Source model may not function well in all business environments. Commercial businesses selling products to the Amateur Radio community, on the other hand, may owe their existence to the hobbyists themselves. While this doesn't happen in all cases, there are many occasions when the devices available were inspired by a Ham or a group such as TAPR, BayCom, or the Ottawa Amateur Radio Club. It is the work and inspiration of many individual people that grant us such wealth in Amateur Radio operation. Many companies were started thanks to such people who had the good sense to add to the community.

Companies can use the Open Source model to allow Hams to help find errors and resolve them. As Hams have done in the past with solving electronic quirks in equipment and kits, now too can they find software quirks. As more and more equipment has the need for computers and embedded controllers, so too the need arises for people to spot software problems. This industry serves a community perfectly suited to this type of open exchange. Rather than the company using resources tracking down some obscure bug, we may find that some Ham has the answer the company didn't see. The company gets an easy fix and more time for other business, and we all get a better product. And as laws have protected companies' electronic designs in the past,so too can new and existing laws protect their software.

## My Project, Xastir

Xastir came about for two reasons. First, I wanted to build a tracking station for my local balloon group. Second, I use Linux and didn't see any signs of getting a typical APRS(tm) program, that is, one with graphic display, maps and messaging. Also by typical I mean with source code according to the Unix/Linux model. If such a program was available, for which I could modify the source code, I would have gladly paid the registration fee and sent my meager additions to the author. Since there was not so much as a binary version on the horizon, I proceeded on my own.

Xastir stands for X windows Amateur Station Tracking and Information Reporting, and is an APRS(tm)-like program. Although it is unfinished, it does fulfill a need and many people in the Ham community are using it. This project wasn't necessarily going to be an Open Source project. I was writing it for myself and my club, but due to the many requests for a Linux version of APRS(tm), I finally decided to make my project available for all to see. It is distributed as Open Source to reap the benefits of this model, both for me and for any other person interested in this project.

This project is fairly young and has been Open Source for a very short time, but I have received nothing but positive response from the community. Many have helped this project along with little things here and there, offering what knowledge or code additions they could provide. Hams from around the world have conveyed interest, all offering what ever help they can. From Germany, a Ham sent suggestions and code to help me change my software to function within a Unix-type file standard. From various locations in the United States, I have received faxes and code pieces for adding Weather Station decoding. In the North West US, I hear that my program has been ported to BSD and Sparc, with progress on a Solaris version. All of these contributions will help me add to my software.

With the source code available, conversions to various languages can be worked on. Unknown types of weather equipment, TNC's, operating systems, and local preferences can be added. It should allow more talented people than myself a base to work from and improve upon.

## Conclusion

Open Source can mean many things to many people. At its core is the idea that with freedom of open code development, benefits will come from interested users and programming professionals. These can affect a project's development in more ways then just the code itself. Time and trouble from bugs in development can be reduced by some members of the community interested in your project. Larger teams can be formed to better test and implement your requirements. Open knowledge can better your project and inspire others to create their own.

APRS(tm) is a Trademark of Bob Bruninga

# Automatic Weather Bulletins via APRS®

Dale Huguley KG5QD
100 Sandhill Road
Marco Island Florida 34145
kg5qd@worldnet.att.net

Keith Sproul
698 Magnolia Road
North Brunswick, NJ 08902
ksproul@vger.rutgers.edu

**Background:**

This project was an outgrowth of a Pascal-based weather parser located at the Collier County Florida Emergency Operations Center called **WXSVR**, which stood for Weather Server or Weather Severe. Data from the GTE Weather Wire service was broken into products and made available on the local packet BBS, with hurricane data sent to the statewide network. In February 1997 I met Keith Sproul at the National Hurricane Center during the annual Amateur Radio Conference. I started communication with him and Mark Sproul concerning the use of WinAPRS™ for hurricane information display and dissemination. An interface protocol was agreed upon to allow the development of a parser as a possible plug-in to the MacAPRS™/ WinAPRS™ software. The Parser was originally for hurricanes only, but subsequently was developed for all types of weather bulletins.

**Weather parser:**

Weather text data is available via Satellite from the GTE and EMWIN feeds, and via the internet. E-mail versions of the products are also available on a less timely basis for testing and backup. The parser is written in 'C' programming language with the input module able to handle any of the available sources with identical output. This can be file based, or as in the case of the GTE Weather Wire , a serial input. The output of the parser is a formatted packet ready for transmission. Output is sent to the retransmission engine as if it were received off air.

**Retransmission engine:**

One of the hardest nuts to crack in this whole scheme is the problem of when and how often to retransmit data to assure it has made it through the system to the desired destination. A standard decay or a rhythmic retransmission does not address the fact that too much data can be dumped blindly into a local system at the very time the system is needed to relay information to the NWS. Also each area will have a different set of priorities for its weather data. Rather than having a central parser for the entire country

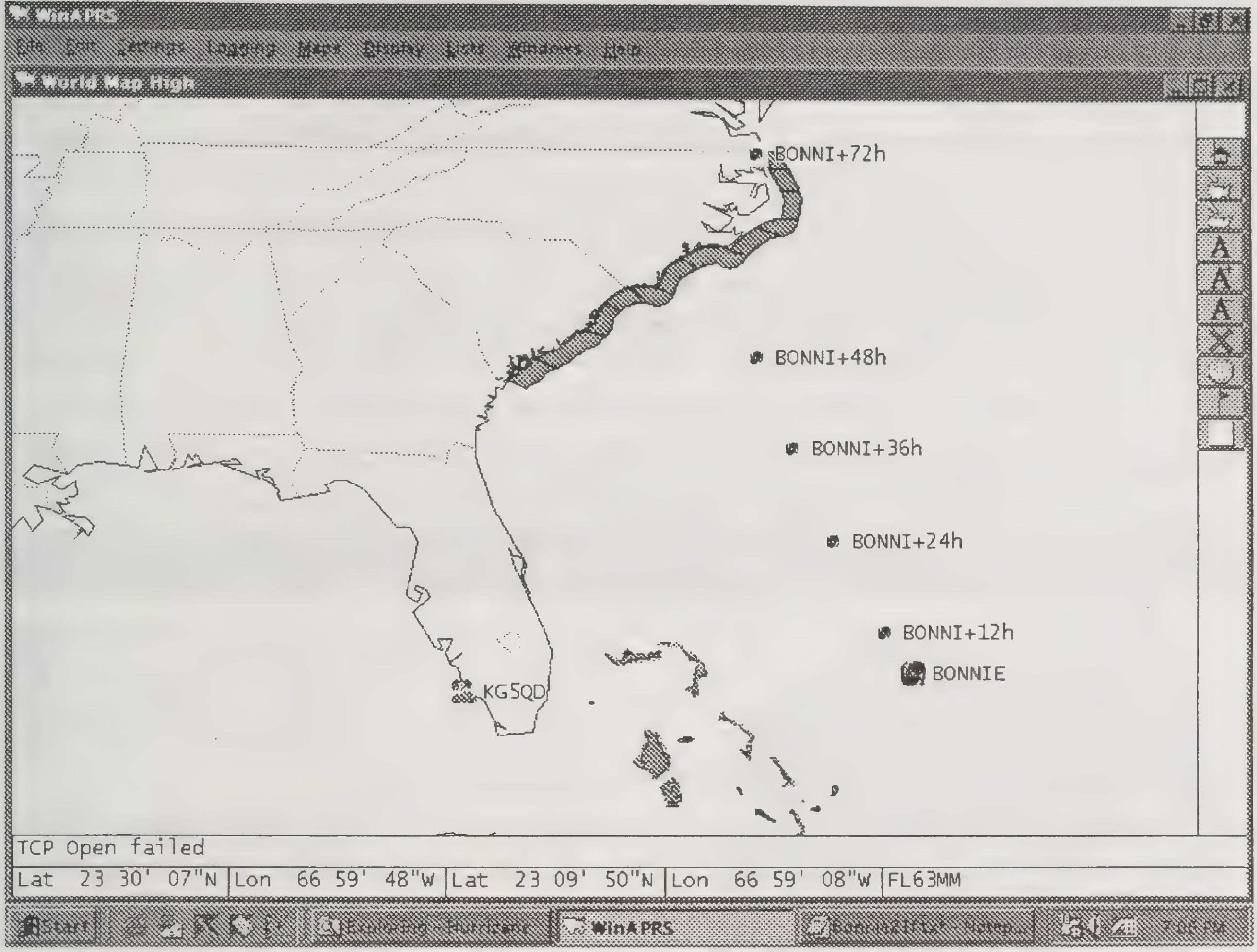
determining these matters, a decentralized approach seems more appropriate. The priorities are assessed before every transmission to determine what is the most important data present, when was the last time transmitted, and by what path it should be sent. Since the retransmission engine is written to accept input from the parser or from any path available, the system is redundant and not reliant on any one input source.

**Hurricane Parsing:**

The Hurricane Center produces a package of bulletins for a tropical cyclone (generic name for hurricanes, tropical depressions, tropical storms etc.). The **forecast\advisory** has an array of present and predicted positions, along with wind and windfield data. The protocol developed by Bob Briuninga, WB4APR, for hurricane plotting allows the most vital information concerning a tropical cyclone to be transmitted in packets of 65 characters or less. It was realized quickly that while the packets generated left the parser in a coherent fashion, the APRS system could cause the packets to become confused with reference to time and order. A naming convention along with a **sequence tag** was developed to allow display software to reconstitute the data into a uniquely definable whole. This was done in line with the already existing message format and numbering protocols.

Subsequent updates for the tropical cyclone will have objects with the same names, but with different sequence tags. This allows old data still in the APRS data stream to be suppressed. Because one of the items parsed is the time of the next scheduled update, it is possible to set the duration of the retransmission of the data and assure only timely data is being transmitted. This requires that the objects parsed not be handled in the normal decaying fashion common to other APRS objects.

Display seen from WinAPRS created by the hurricane data and coastal zone alerts created by this software.



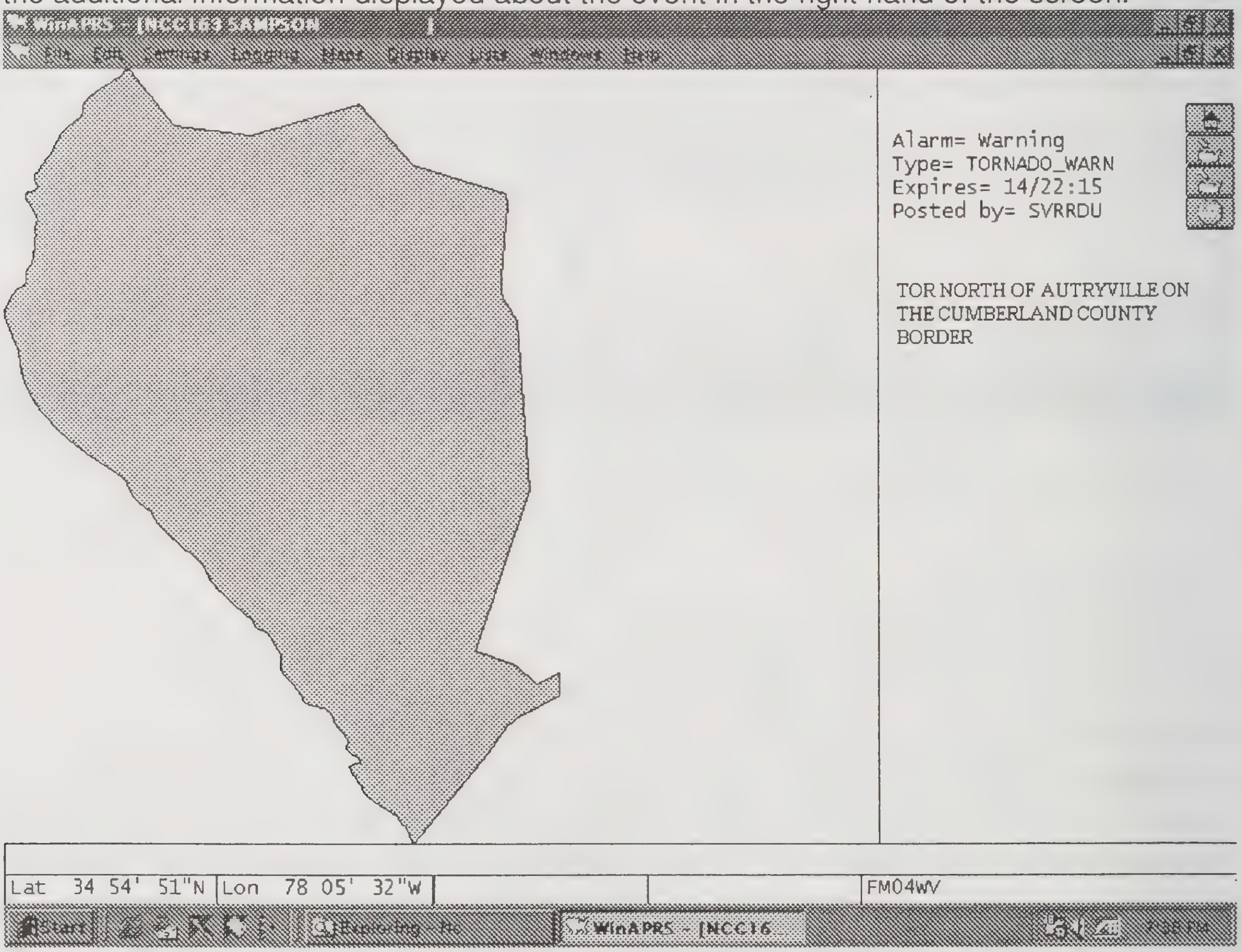

**Weather Bulletins:**

The Sproul Brothers developed a manual mechanism for issuing weather alarms, warnings and watches via a graphical display based on county outlines. This system causes the counties to high-light in different colors, depending on the type of watch/warning etc. This system caught on very quickly and is being used all across the country for issuing county wide weather watches and warnings. There are approximately 3100 counties in the US and all of these files are currently available to the WinAPRS/ MacAPRS/ X-APRS user. APRSdos displays these watches and warnings slightly differently, but it does display the data.

We have now taken this concept and created another 3500 inland and marine zones based on National Weather Service shape files. These area files are what the automatic weather parser issues its warnings and watches for. The nice thing about this is the APRS user only has to get new 'county' files and does not have to get a new version of software to make this work. Therefore it is fully backwards compatible for the last couple of years.

In addition to the NWS bulletin that causes the appropriate county or ZONE to be high-lighted, the additional information now gets 'associated' with this event and will come up on the information screen if you double click on the high-lighted county.

Weather Warning screen displayed when you double click on a high-lighted county. Note the additional information displayed about the event in the right hand of the screen.



It has been determined that in order to associate packets (which may arrive in any order) and assure that there is no ambiguity in this association, each packet needs to contain the following:

1. National Weather Office that issued the bulletin
2. Type of bulletin
3. Time the bulletin was issued

In practical terms some compression of this data was needed, as the length of packets are limited to 65 to 75 characters. Each NWS office has responsibility for a County Warning Area (CWA), which has a unique three letter code. This code is embedded in the packet header. This allows the packets to be steered to the appropriate area of the country using a built-in feature of I-gates. A single character type of bulletin along with a three character code denoting time-of-origin and a single character line number is added after a curly brace character to each packet. This is fully compatible with the message numbering system already in place in the APRS system.

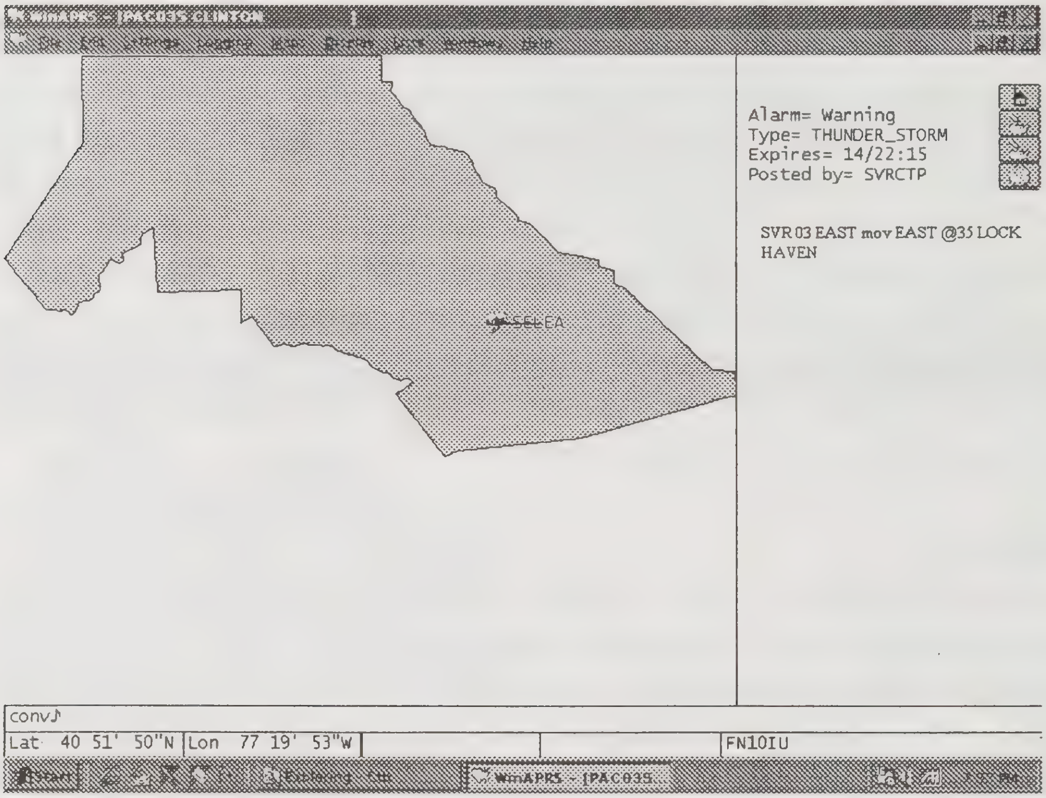
**Location Information:**

In the process of parsing what the NWS calls **short fuse** warnings, which are tornado, severe thunderstorm and the like, it was discovered that most of the time the text concerning the actual location of the storm could be reduced to a single name, offset, course, and speed. This makes it feasible to display the data on a Kenwood TH-D7A handheld. Furthermore, it is possible to plot the actual location of the storm as a regular APRS object due to the fact that the names used for locations by the NWS are from a known database.

Actual data transmitted over the air

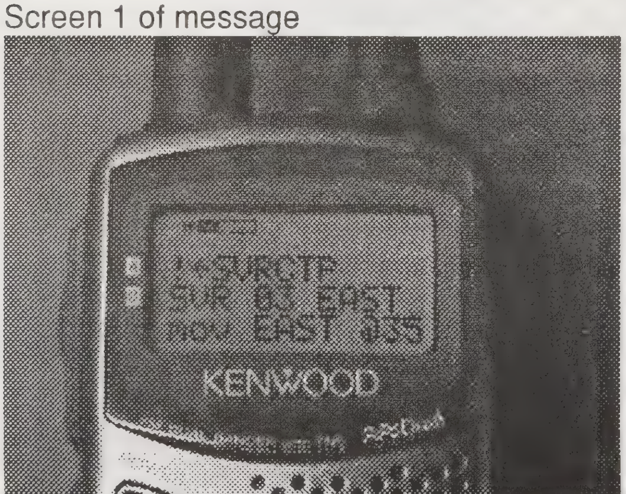
```
SVRCTP>WX:NWS-WARN   :142215z,THUNDER_STORM,PA_C035,    {SELEA
SVRCTP>WX:NWS-SELEC:SVR   03  EAST  mov  EAST  @35  LOCK  HAVEN
```

The data shown above created this on WinAPRS/MacAPRS screen.

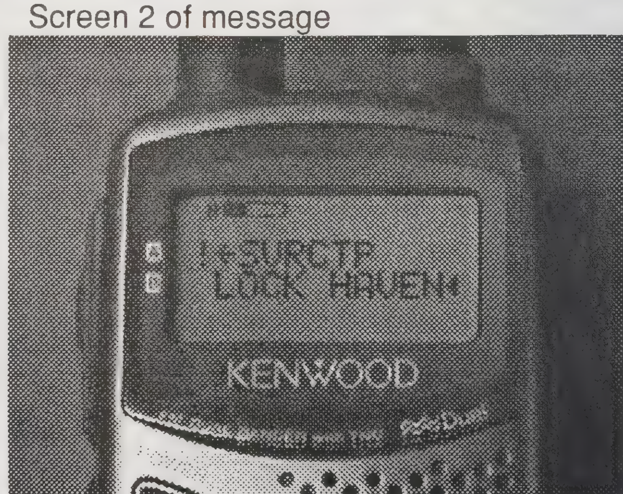



The same data created this on the Kenwood TH-D7A.

Screen 1 of message




Screen 2 of message



**Conclusion**

APRS has developed into a very useful weather tool. We also now have the ability to have data from many different sources fed into APRS and this data can be redistributed out to those that need it in a timely manor. This addition to APRS is providing the ability to take the complex information from the National Weather Service and strip it down to what is really needed in the field. This information can be displayed in a useful manor on computer screens and also on the new generation of portable APRS stations such as the Kenwood TH-D7A.

# (Non Technical) Lessons to be learned from the PSK31 Phenomena

Eduardo Jacob, EA2BAJ
ETSII e IT, Bilbao
Dpto. Elect. y Telecom. – Ingeniería Telemática
Alda Urquijo S/N
48013 Bilbao SPAIN
jtpjatae@bi.ehu.es

**Abstract**

The new PSK31 mode has raised much attention from both the technical press and the hams. We can get on the air from many operating systems using different hardware. We can read about it in many languages, ranging from English to Czech. There have been tests on satellites and on high frequencies. Many contests now include PSK31 as a valid mode. Yet, PSK31 has much more to offer. I believe that we can learn from the experience in benefit of new ham projects for the future.

**Introduction**

As some of you may know I spend some of my ham time maintaining the PSK31 WWW Page on the Internet [1]. For that reason, I have followed the PSK31 evolution from the early days of a unique version for DOS on the Motorola 56002 EVMDSP up to this days when there are many versions for different hardware and software platforms. For many reasons, it has really been a fun time that I will probably not forget easily.

I believe that, independently of the final status and impact of PSK31 within the Ham community, there are many lessons that can be learned and put in practice in other projects. This paper will leave out technical details, for there are already many papers, articles and pages on the Internet about this.

**Who did the job?**

As you may know PSK31 is the brainchild of Peter Martinez G3PLX. He alone has designed and coded, the first implementation of the mode. I think that this has been possible not only because of the great amount of time that he has been able to put into it, but more notoriously because of his own capacity.

Today we can see big pieces of successful free software being the result of the cooperative work of many individuals either conducted by a person (as Linus Torvald with the Linux Operating System) or by a team (as in the Apache WWW server or FreeBSD Operating System). In this context, some people may argue that the approach taken here is by no means democratic at all; in my opinion I think this is a simplistic reduction of reality. First of all, the work (if for free) must be done in accordance to one's own likings as it's the only way to get results. In second place, design issues have been clearly explained in documents given away together with the program, thus provoking feedback between potential users and the author who has been receptive to suggestions and opinions, although retaining the right to consider them or not. In some way this means that decisions tend to be public and shared in

every stage of the design. This takes out "magic" based decisions. In third place the project by itself still fits in the work a person can undertake by himself, as we can see in other PSK31 implementations that are seeing the light lately. This shows us that this approach hasn't been a brake for the PSK31 mode. A small team (less than 3 or 4) could also probably use the same approach even if the entropy clearly increases. It could be a solution if less time is to be dedicated to the project by components.

The conclusion is: *The one-person approach is suitable to produce public consumption results if some care is taken to share design decisions with a mind open to suggestions, and if the work fits in one's person's capacity.*

How do we validate this work? The answer is with an appropriate group of testers (in other circles alpha/beta/gamma/release candidate/gold/xxx testers). In PSK31 the tests conducted have been very important in the definition of both performance and usability results. The very early tests where conducted using a (at that time) not very deployed platform (the Motorola 56002 EVMDSP). The people who were using it at that time were of the kind that can concentrate on testing the mode instead of wasting time with setup or cabling problems. So although the testing of the early versions was open, the public addressed was at least limited, so the final result was that a small, active and qualified group did the testing. I don't remember the exact number but I don't think there were more that 15 persons involved in it. It's important to say that testing is more than just installing and trying once a program. If you find something wrong, you must try to locate the bug (if there is one), and how to reproduce it. It's a hard work.

Although I initially setup the WWW site and it's mailing list for supporting tests, curiously enough the major part of the feedback was sent back to Peter using direct email or PSK31 QSO's in the 80 meters evening roundtables we had. This is clearly opposed to the experience I have had in other tester lists. The list was mainly used for announcing new versions that could be downloaded from the site. This was probably caused by the fact that almost all the testers were located geographically near and could attend the roundtables.

To sum up: A *small, active and qualified group of testers is needed to conduct on the air and usability tests. The natural selection (due to the hardware needed) that we had at that time, should be probably substituted by an artificial selection. The Internet proved to be interesting to propagate new versions, and the feeling is that had the test been conducted in a wider area it would have been interesting also as feedback channel.*

**Design issues.**

Peter designed the mode with some clear objectives in mind as described in [2]. Later he expanded the initial design to include another modulation and a broader set of characters. It's important to note that all the improvements were made in a compatible way so the initial design showed to be expandable. The design description was included in the package. The need to have this description available as a document was detected and as a result a page was added on the WWW site giving detailed technical information about it.

He also proposed and produced a free working implementation reference. This has proved to be of capital importance. First there is a free reference that can be used to test the design of other programs,

free or commercial. Second, it gives people the opportunity to test at, no cost, which are the requirements, the operation, and the feeling of the mode.

The code for the algorithm although not freely available has been supplied to third parties that were interested in including this mode in freeware only packages.

The program was not in any case intended to be the "Ultimate PSK31 auto-bragging-contesting machine". This took some time to be understood by people and is still a subject that resurfaces periodically in the list.

Perhaps the fact that the implementation reference was free pushed volunteers to translate the help files to different languages. I am not sure of this because we can see that there are shareware and commercial programs that also have translations made by volunteers. In any case, this has been very important to the propagation of the mode.

The conclusion is: *A good, detailed and expandable design, freely available, along with a free implementation reference have been instrumental to the springing of the new breed of PSK31 hardware and software and to the diffusion of the mode.*

**The hardware and software platform elected for the implementation reference**

Early in the testing, it was clear that the first hardware platform selected was not going to be the definite one, because of the price and complex setup. Later Peter produced the Windows © and SoundBlaster © version. This meant that the price was not anymore a problem, because the computing power needed was very small and the soundboard required was very common and cheap. The issue of not being using a free operating system didn't stop the propagation, but at least caused a Linux with soundboard version to appear.

The hardware needed to link the soundboard to the rig is very simple, there are kits available that offer more than the needed functionalities, and it is incidentally the same one that is used for other SB-based applications as RTTY and SSTV.

The application itself should not be considered as a state of the art Windows program that uses all the resources available for GUI design, but a proof-of-concept instead. This also led to some comments. The only problem that has appeared is related to compatibility issues with "SoundBlaster compatible boards".

I think we should not forget to mention the solution given to one of the difficulties of the mode: tuning/frequency accuracy. Before Peter added AFC and the waterfall display, it was very difficult to hook up with somebody. This very technical approach to the problem and its solution made possible, even for the uninitiated, to participate. This clearly made of PSK31SBW a "killer app."

In a few words: *The wide base of users PSK31 has got is mainly due to the selection of a low cost and widely available platform, (Windows and © SoundBlaster ©) along with a very usable program that concealed inherent mode difficulties.*

**A place in the band**

One of the points (if not the only) that caused frictions was the election of the calling frequencies. Initially, Peter proposed more than 3 years ago to concentrate activity starting from the bottom edge of the IARU RTTY bandplan, expanding upwards as activity increased. The exception is in the 10mts band, in order to give non-full privileges ham an opportunity to meet. It was defined as 150 Hz above it in order to sit between existing Pactor mailboxes. This approach is not new as it was used when AMTOR appeared and proved to work. In IARU zone 2, other frequencies were selected leading to both confusion and eager comments.

The problem could be explained, by alleging that there is no official room in the bandplan for experimentation. The truth is that perhaps nowadays there are not sufficient experiments to justify it. Also, this wouldn't solve the problem of how do we add a new (no experimental) mode to the bandplan. Indeed, our experience is that we almost hadn't problems during the test, and that these appeared with the general utilization of the mode.

Almost every user is jealously watching and protecting the slice of the band he is using and takes it as a personal affaire to enforce the "right" use of it. So, we have seen carriers of many types going up and down to iron PSK31 signals. This doesn't mean that PSK31 carriers have never irrupted on other type of established QSO's. I think this truly reflects the problems we are already aware of in other modes, with the novelty that perhaps on the first times, the "warbling" wasn't associated with a proper mode.

In zone 1, on 20 meters (the more problematic band), the approach of explaining to Pactor mailboxes operators the problem and asking them to move a little up in the band, has been very successful, and is the recommended way to pursue a common "PSK31-land".

I have personally missed some expression of interests from the official organizations, at least in EA-land.

The conclusion I extract here is: *When a new mode appears, much care has to be taken in order to define the calling frequencies. The approach used (150 Hz above bottom edge of IARU RTTY bandplan) as been very successful, at least in Zone 1. In any case, different parties proposing different frequencies only lead to confusion and sterile discussion. Facing the problem and speaking about it with involved parties is certainly the way to go.*

**Diffusion of the mode**

I dare to speak about the Internet, for many the end of the Ham activity. I think that we can talk about it many hours, most of them for nothing productive. In the PSK31 case, the Internet has been very important for the diffusion both of binaries and information.

The Web Page has been the original point of distribution of several programs including the reference version by Peter. Much care has been taken in order to give as much information as possible about the mode, including pointers to other software or hardware platforms, articles, and other related resources.

The mailing list that was initially created for testing purposes was later recycled as a general use mailing list. Several features have been added later at users request, as digest-mode, mandatory use of

text only messages, searchable message database, etc. Other mailing lists have appeared in many places around the world.

The fact that I administer the servers where everything is located allows me to give other kind of information. Overall, the feeling is that the resource can be employed better. There is an archive of old messages, which is not very used. Many people ask questions that are already answered. The same thing happens with the WWW site when they search information about versions, etc.

Many users still don't understand what is behind the nice Internet interface modern operating systems offer. There are many problems with subscribing, unsubscribing and using multiple aliased email addresses. The "free email address of the week", which is free just that week, or that only allows a few kilobytes in the mail spool also is the root of many problems.

Other media, as magazines, or local reunions have also helped the diffusion of the mode, especially for "unplugged" hams. They have also been instrumental to bring attention on PSK31 and to put the seed for the Internet access to resources. The effect of the article that appeared in May'99 QST caused an explosive use of the mailing list and WWW server.

As I previously said before, there are translations of the helpfile and many articles in different languages. I think it would be interesting if authors ask permission to editors to republish the article some time later in the Web.

In another order of things, many users don't read the documentation at all. Peter explained once that a –great– amount of effort had gone to produce a good help file and people seemed to be ignoring it. There is a bad habit of using the Internet (asking in the mailing list, or directly) instead or personal reading or studying. Many users don't realize that it is not ethical to ask others to spend more time and effort answering them than the one they have used searching and asking.

For example, the problem of overdriving is clearly explained everywhere in the helpfile and there are many messages about it in the list. Still, many people don't pay attention to it and follow the approach of "don't touch it if it works" even if they are kindly told to correct it.

The lesson here is: *The Internet has been very valuable for the diffusion of PSK31 as a point of distribution of both binaries and information. We still have to increase our skills and knowledge to fully take advantage of the power of this medium. Other media as magazines and local reunions, have still an important place as vectors for pointing information to both connected and unconnected people. Users have to fight against their laziness, and culture experimentation even if now it involves only ready-made software.*

**Conclusion**

PSK31 is probably one of the biggest events of the last times. I have presented some not technical lessons extracted from the process that brought us this mode. I think that they can be useful if other individuals or small group of persons face another project of this kind. These points can also be detected in other successful projects we find around. Finally the PSK31 phenomena fits the charter of amateur radio well, i.e., the means and ability to use the hobby for personal education, and it complies also to the open software philosophy that is becoming the way to do these things.

## Reference

[1] The PSK31 WWW Homepage http://aintel.bi.ehu.es/psk31.html
[2] Peter G3PLX, PSK31: A new radio-teletype mode with a traditional philosophy. Radcom Magazine, December 1998 and January 1999, available at http://det.bi.ehu.es/~jtpjatae/pdf/p31g3plx.pdf

# Arizona Packet Radio, Past Present and Future (?)

Keith E. Justice, KF7TP
6759 Wagonwheel Lane
Lakeside, Arizona 85929
kf7tp@cybertrails.com

Daniel J. Meredith, N7MRP
P.O. Box 6687
Concord, California 94524-1687
dmeredith@phx-az.com

## Abstract

A brief history of packet radio in Arizona is presented. The current status of the network is described with a map showing node locations and major links, and a detailed node list is available in the Appendix. We speculate on the future, arguing that the Internet, while responsible for the decrease in the current user population, can also provide opportunity for future applications. An application particularly suited to Arizona is the provision of Internet e-mail gateways for the many vacationers, winter visitors, and campers who frequent the state. We are optimistic that other applications, not foreseeable in their exact nature, will certainly emerge.

## Keywords

Packet Radio, Arizona, Internet, Gateway, E-mail, BBS

## Introduction

Despite having been the birthplace of TAPR, the TNC1 and TNC2, Arizona has not been host to a particularly well developed packet network. While TAPR evolved into an international organization with strong leadership and a membership with rich technological skills, the local user organizations never quite achieved the organizational capacity to put up a broad high capacity network. Nevertheless, some early organizations made substantial contributions to the Arizona packet network. This paper describes briefly the history of packet radio in Arizona, provides a snapshot of the present, and speculates on the future.

### The Past

A group in Scottsdale who founded the Arizona Packet Radio Association (AZPRA) in the 1980's initiated the first statewide organization. AZPRA succeeded in linking the two major metropolitan areas, Phoenix and Tucson, with a 9600 baud backbone connection on 220MHz. The two sites linked were Mount Lemon, serving Tucson, and White Tanks Mountain, serving Phoenix. User access to these nodes was at 1200 baud on different simplex frequencies. As is often the case with our mountain top

user nodes, the simplex frequencies offered links to other mountain top nodes in turn. These adventitious connections were a mixed blessing because long-haul traffic on the user frequency often led to congestion and slow response time for local keyboarders.

The Maricopa County Repeater Group made another notable effort in the early days. This small organization succeeded in establishing a remarkable array of 1200 baud user nodes and 4800 baud, 6 meter, backbones stretching from east to west across the central belt of the state. User nodes were on Greens Peak near Springerville, Pinal Peak near Globe, in Phoenix on the Valley National Bank building (the tallest building in Phoenix), on Mingus Mountain near Prescott, and on Hayden Peak near Kingman, Arizona. The backbone continued to Mt. Potosi near Las Vegas, Nevada, and Big Bear Mountain in central California, permitting keyboard access to the Los Angeles basin (See Appendix, Attachment 1). The nodes on Pinal Peak and in Phoenix were regenerative duplex repeaters, thus avoiding the "hidden transmitter" problem. The Phoenix node continues to provide reliable service to the metropolitan area, with the original hardware!

The first 9600 baud user activity in Arizona grew out of the need for the TCP/IP users to get away from the ax25 community and achieve higher transfer rates. There was considerable friction between the AX25 users and the TCP/IP fans in the early years, leading to the construction of a separate network over a portion of the state. This was unfortunate, since the effort could have been better spent in a cooperative effort, but it seemed unavoidable at the time. 9600 baud users are still almost all TCP/IP oriented, but the network they built is now broadly used for mail forwarding, ax25 keyboarding, DX Cluster backbones, etc.

Another factor that helped shape networking in Arizona is the DX Cluster. As with TCP/IP, the early cluster traffic rode on existing nodes, but with the potential for channel overloading at times, the DX community proceeded to build there own network. More recently, some backbone links and user nodes are being shared between DX Clusters, TCP/IP, BBS forwarding, and AX25 users.

In recent years, another organization, the Arizona Network Intertie Group (AZNETIG) did establish an effective backbone with several user nodes in the central part of the state (See Appendix, Attachment 2). In 1993, Daniel Meredith, N7MRP, founded AZNETIG with the former assets of AZPRA and the goal to revitalize packet radio, as well as attempt to bring together the "movers and shakers" with regular coordination forums. The first such coordination meeting occurred in Casa Grande and the turnout was unprecedented for the State. The entire State of Arizona was represented at the meeting and the beginnings of frequency and idea coordination were finally underway. Meetings have continued throughout the 90's, though participation and excitement has waned as time has progressed. Regional organizations, formal or informal, in Tucson, southeastern Arizona, the White Mountains, and elsewhere also made significant efforts to provide connectivity in their areas. But many of these efforts were dependent on one or two individuals who carried the ball to the goal line (or the radio to the mountaintop).

While there was activity surrounding network nodes and links, the BBS scene was off and running, pushing packet radio technology to the limits. The sudden flurry of activity began in early 1992 with the introduction of low cost TNC's and radios that offered plug-n-play packet interfacing. The excitement caused network operators to expand and upgrade the networks, particularly the creation of additional backbone paths at 9600 baud to handle the BBS and user traffic. The Phoenix area had so much activity

that at one point there were a total of 6 full-time, full-service BBS's. Each of the BBS operators claimed stake on a portion of the user share by offering multiple frequency access, which included HF, VHF, MVHF, and UHF. The strain on the networks from the BBS forwarding forced the BBS and network operators to begin working hand-in-hand, rather than as individual entities. The BBS operators became integral to the networking process and were involved in all networking meetings taking place around the state. The result was an agreement to forward NTS and personal traffic 24 hours a day, yet bulletins would only be forwarded during the off-peak hours of midnight to six o'clock in the morning. In addition, the BBS operators agreed to fully support the networks that were used to pass traffic, thus relieving much of the maintenance and financial burdens often associated with mountaintop radio sites. As time progressed, so did the software the BBS operators used. The introduction of the French "FBB" BBS software provided compressed forwarding, as well as many customizable features allowing BBS operators to differentiate their systems. Notably, Phoenix also became the home of the F6FBB Support BBS for the United States and also served as a telephone gateway to packet radio in Arizona via N7MRP's station. Times have since changed and with the advent of the easy access to the Internet, the bustling BBS days have all but disappeared in Arizona, leaving very few stations for users to access.

Unfortunately, Arizona has never seen the emergence of organizations such as the Texas Packet Radio Society (TPRS) or the NorthEast Digital Association (NEDA). Perhaps one reason why Arizona seems to be organizationally challenged is that our "basin and range" topography makes long range communication easy for individual node operators to achieve. We do not need multiple hops to talk across a hundred miles. One ham can put a node on a mountain top, serve a lot of users with it, and tie to another mountain top 50 or more miles away with the same 145.01 MHz frequency, the distant node having been put up by another individual ham or informal group of users.

This obviously is not good engineering practice, because it produces a textbook example of opportunities for the "hidden transmitter" problem. For example, the LMN node is at an elevation of 9157 ft. on Mt. Lemon in the Santa Catalina range north of Tucson. It sees much of Tucson as well as the valley surrounding Phoenix, and the small towns in between Phoenix and Tucson. It also talks to other mountain top nodes such as BISBEE ( 78 mi.), SONORA in Mexico (117 mi.) , JACKS in New Mexico (125 mi.) , and UNION (156 mi.), and even sees ELDEN, 200 miles to the north, although that path is not usable. Some of these nodes see each other as well as LMN, but obviously, nodes to the south of LMN do not see those to the north. The several ground level nodes that LMN sees, and many of the users, do not see each other. Under conditions of light traffic, LMN works. When congestion begins to build, not much gets through.

The current or past sysops of these mountain top nodes cannot be faulted for this situation, for before these nodes came into existence there was nothing. We do not mean to "point fingers", since most all of us who have put up mountain top nodes have, at one time or another, contributed to the problem. The solution for user nodes is to go to full duplex regenerative digital repeaters on mountaintops, which is easier said than done.

In the recent past, the APRS community has installed a number of digipeaters in Arizona. We leave the discussion of that activity to David McCarthy in another paper to be presented at this conference.

In the face of all of these struggles to maintain a packet environment in which all classes of users could achieve their goals of having fun with packet radio, Arizona nevertheless had a growing population of

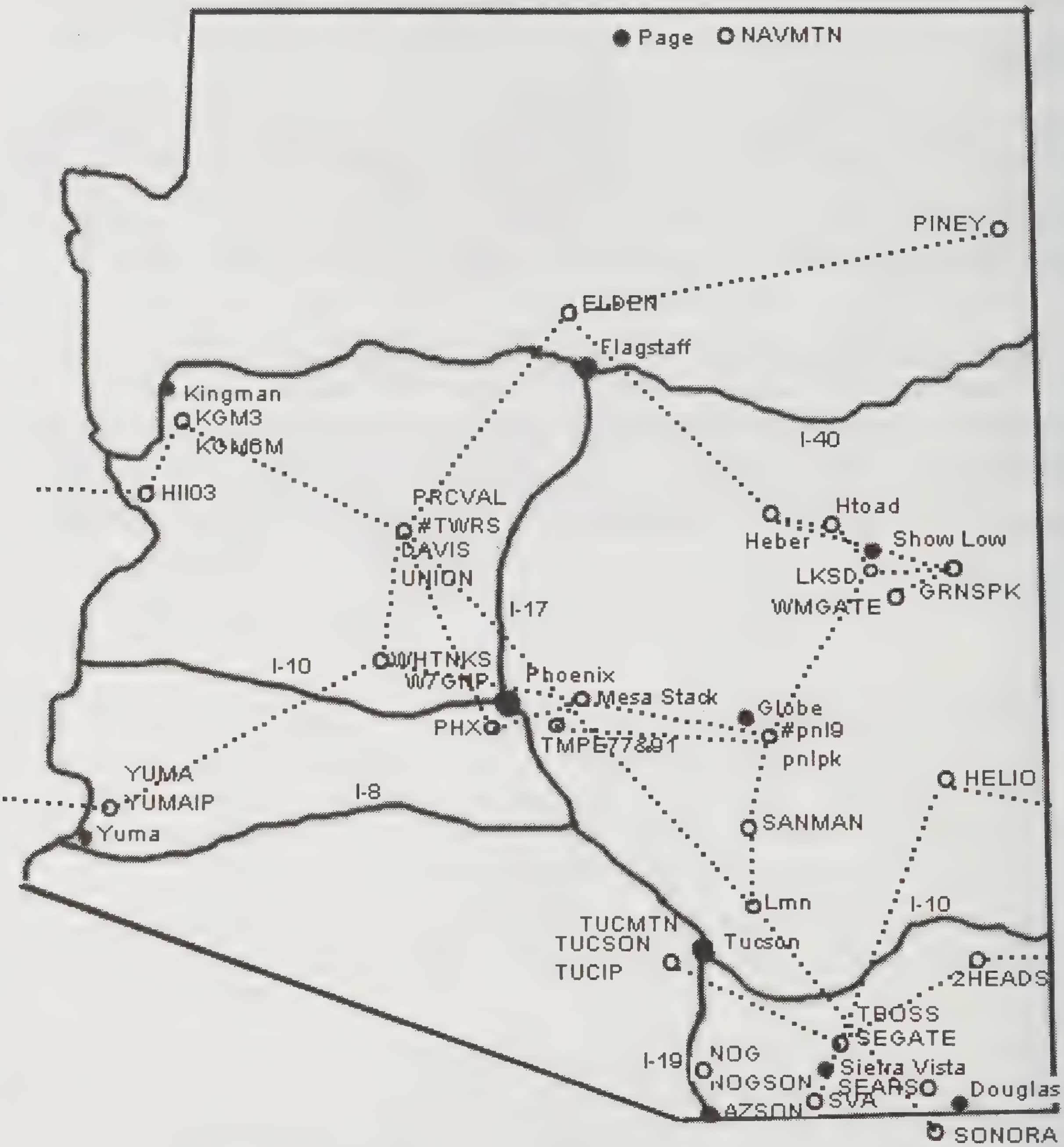
users until the infamous Internet arrived! In the blink of an eye, our problems with congestion disappeared. Even some of the sysops lost interest due to the dearth of users for their nodes and bulletin boards. One new dimension was added to packet radio, namely, the Internet Gateway/Wormhole, but the general packet user population still lags far behind where it was a few years ago.

**The Present**

The packet network in Arizona is presently a patchwork quilt that works for the dedicated practitioner. Some of the backbones are broken, replaced by internet wormholes, some of the user nodes are broken or missing, and there are no user nodes over 9600 baud in speed. But we are optimistic for the future, as described in the next section. Figure 1 below shows the location of nodes and some of the long haul radio links. Many of the details had to be left off the figure for simplification, but a complete list is provided in the Appendix.

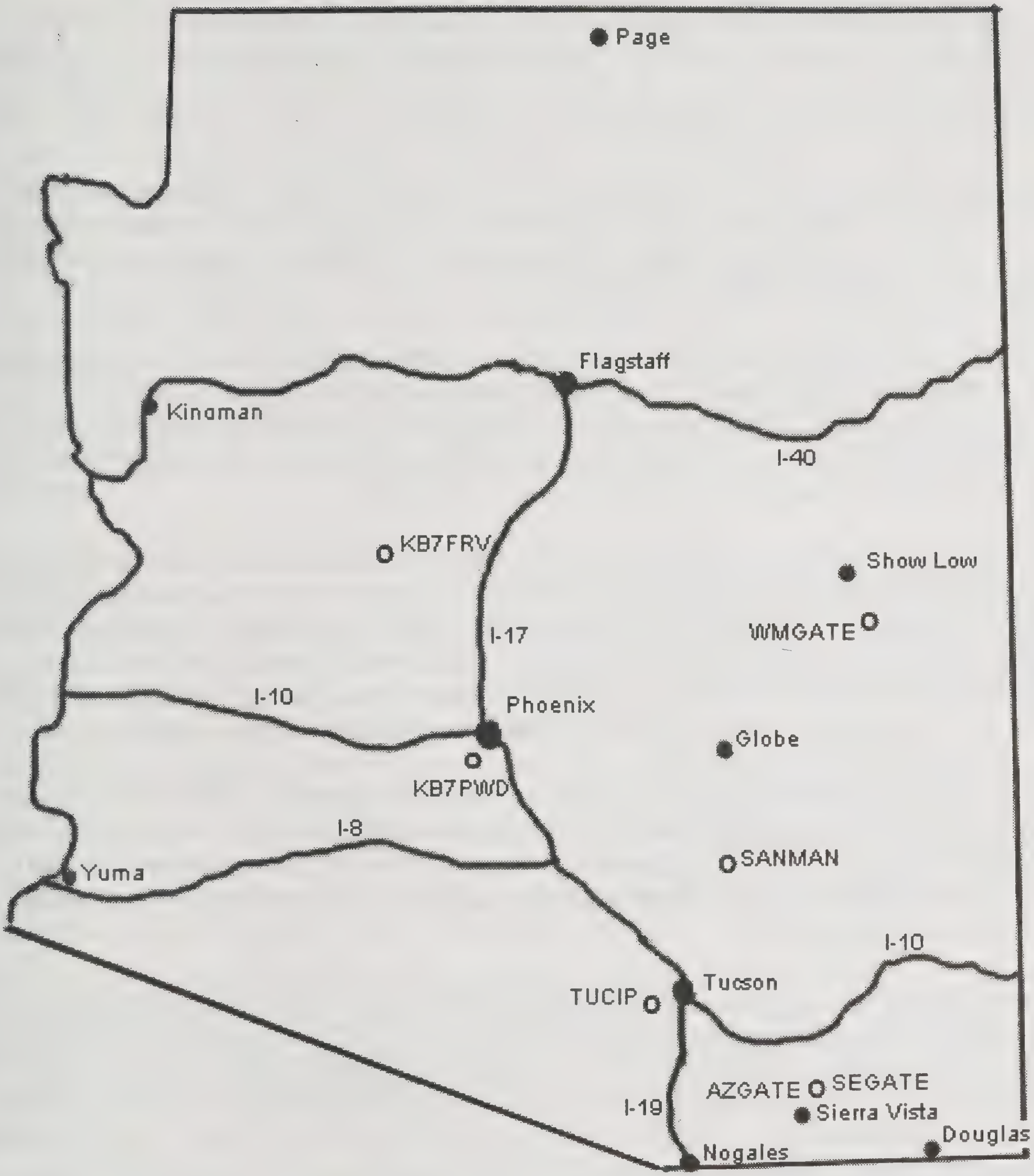
Most of the long links shown in Figure 1 unfortunately are NOT on dedicated backbones. For example, the current links from LMN, as described earlier, are all on the 145.01 user frequency, although a previous backbone linked LMN with WHTNKS and improvements are in the works. In reality, many of the long links are now made by Internet wormholes. Figure 2 shows the locations of these gateways. While we see these gateways as valuable resources for forwarding e-mail to and from the Internet (as discussed below), we would like to see our long haul links reestablished and increased in bandwidth.

**Figure 1**




Packet Radio Nodes and Selected Links in Arizona

**Figure 2**




Packet Radio Internet Gateways in Arizona

**The Future**

Most node-ops and network gurus have voiced despair at one time or another that packet radio users will ever again return to the medium in numbers sufficient to justify repairing or improving the current infrastructure of nodes and backbones. We believe packet radio does have a future, and offer the following observations for our optimism.

Although the commercial world offers increasing amounts of wireless digital capability for a price and in pre-packaged form, there will always be hams who want to "roll their own", either because they think it is cheaper, or to satisfy their creative need to do it differently. We see several applications where packet radio can play a role alongside the Internet.

APRS is of course a shining example of an application, which suffered no competition from the Internet, and which can in fact benefit from the geographical breadth offered by integration with it. As more functionality is added to APRS (for example, short messaging), and specialized hardware is developed (witness the new Kenwood HT/TNC) we expect the APRS user community to expand greatly. The burgeoning of APRS is not speculation for the future, it is here now!

The next application we think has immediate potential for expansion in Arizona is e-mail gatewaying. This requires no new technology and little in the way of network improvement. As we all know, Arizona is a prime tourist and seasonal visitor destination. Besides the influx of short term tourists during the summer vacation period for destinations such as the Grand Canyon, the White Mountains, and the Flagstaff area, we have a veritable army of winter visitors arriving each fall from the northern U.S and Canada. As more and more of these travelers adopt e-mail as an important medium for communicating with their friends, children and grandchildren, the hams among them will jump at the chance to use packet radio to launch e-mail when they are on the road or settled into winter quarters. Another group that will use this resource are the locals who head for the high country to camp out in the summer time. Finally, we have a small but significant number of permanent residents who live so far out that they have neither electricity from mains nor telephones. Some of these are hams that use packet to send and receive Internet mail.

An example of how this can work is provided by the HEBER BBS in the White Mountains near Show Low. This is an area with many summer homes, RV parks, and USFS campgrounds. The area is served by a full-duplex regenerating digital repeater on Greens Peak (GRNSPK:KG7BZ-2 145.13MHz -600). The node is sponsored by the Navapache Electric Co-op, which also provides the Internet Gateway facilities. George Strickroth, WA3PNT, is the sysop at HEBER, GRNSPK and WMGATE are tended by August Johnson, KG7BZ, and Dave Epley, N9CZV, is the liaison with Navapache Electric.

To send Internet mail via this system, a visitor just connects to HEBER and addresses mail in the usual way. A rewrite file at HEBER detects that this is neither ampr.org nor packet BBS mail, and rewrites the address to route the traffic through WMGATE, the local gateway. When the mail arrives at its destination, the reply address shows as "username"@heber.ampr.org. When the recipient uses the "Reply" button, the MX record at the ampr.org DNS at ucsd.edu routes the message to the WMGATE SMTP server via the Internet, and WMGATE sends the traffic back over the radio link to HEBER via GRNSPK.

Thus all a visiting ham has to do is send out a message to his or her correspondents informing them of the temporary address and advising them not to send pornographic or obscene material, or long attachments. So far, there have been no problems of this sort other than an occasional long attachment sent by accident. Because of the temporary nature of most of these activities, there has little problem with commercial spam.

All we need to serve these needs are good BBS systems reachable from the areas of the state frequented by winter visitors, vacationers, and campers (in other words, just about everywhere), reliable well-connected gateways to move the traffic, an easily-mastered user interface, and a modest public relations effort to get the word out to the amateur public. We believe that if the amateur public came to know they could depend on sending and receiving Internet e-mail from just about anywhere in Arizona, we would see a resurgence in packet radio use here that would carry over to other packet application as well, and would be a credit to amateur radio in general. We think this is readily attainable in Arizona, and in fact it largely exists. As shown in Figure 2 above, gateways are scattered throughout the state, and most include or are associated with BBS systems. We have not checked to see how many of them are set up like HEBER, but certainly they can be. Our task for the immediate future should be to determine how many of these gateways permit easy e-mail accessibility to the Internet, and promote their utilization by visiting amateurs.

Extending the idea of Internet e-mail gateways to web content as well, we speculate that the availability of high speed spread spectrum radios, such as currently under development by TAPR, will ultimately provide Internet access with enough bandwidth to permit web surfing from your RV or tent. Considering that this is a little "ham" in everyone, certainly someone will want to surf the internet from a tent, and at least it will be via amateur radio rather than a commercial carrier. Our challenge then will be to provide high bandwidth nodes for users to access the Internet and each other.

Finally, we predict what we cannot predict. That is, we think packet radio will generate some new applications that we, at least, cannot foresee at the moment. "If we build it, they will come." The past has shown that new and emerging products are often developed out of Amateur Radio's advancement of the State of the Art. We look forward to witnessing a continuation of this process in Arizona, as we saw with the emergence of TAPR and the TNC in Tucson, years ago.

In summary, in about 15 short years we have seen the birth, adolescence, and some would say, the early passing of a new amateur radio mode. We maintain that the news of its demise has been grossly exaggerated. The suspected villain, the Internet, may yet turn out to be a good vampire, if we just have the courage and stamina to enter the doors it opens to us.

**References**

Bradbury, Jim, WB5ACL (1991, September 21). Western United States Amateur Packet Radio Network Map.

Meredith, Daniel, N7MRP (1997). Arizona Network Intertie Group: Arizona Packet Map.

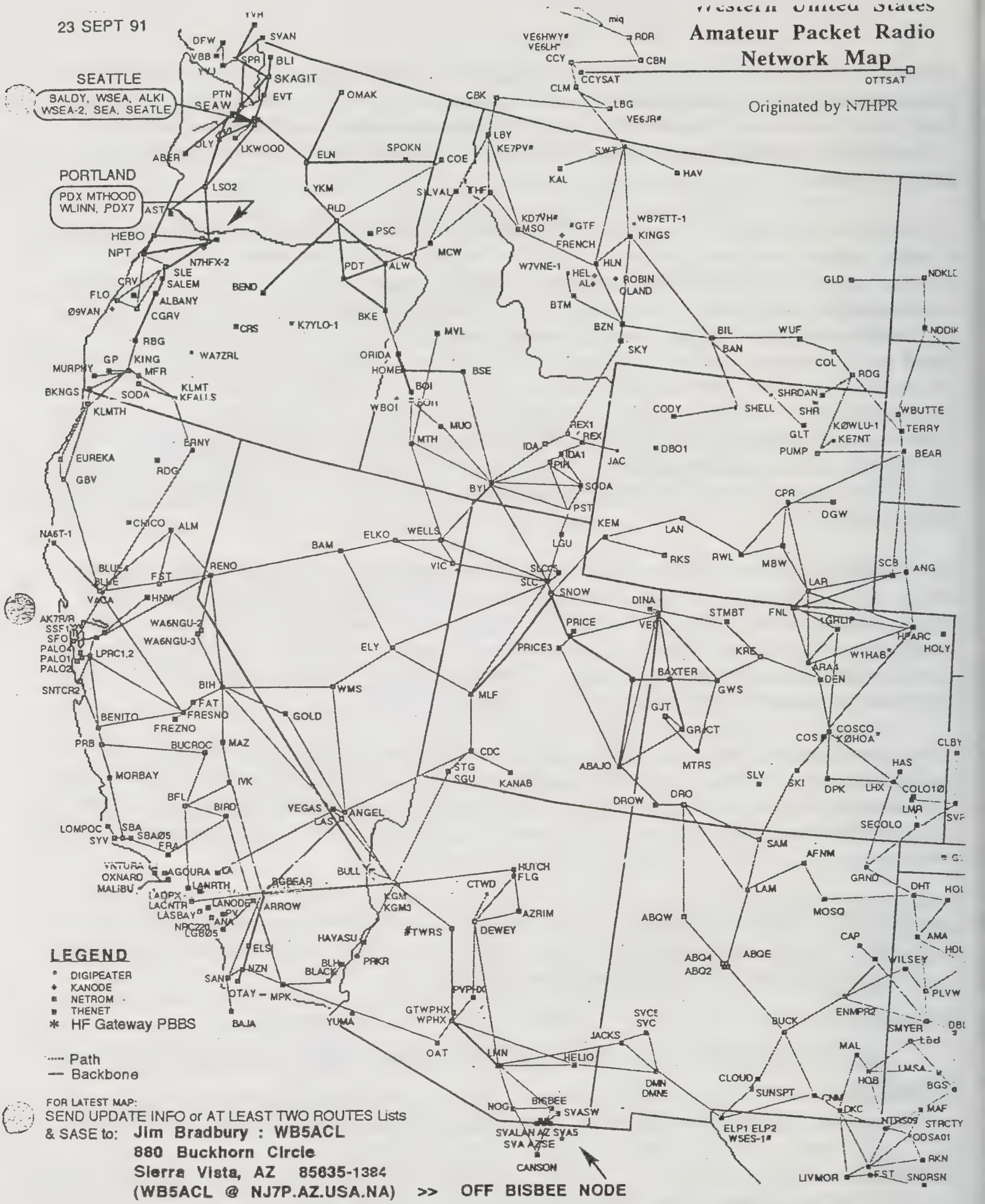
**Appendix**

Attachment 1 - Packet Map of 1991, WB5ACL
Attachment 2 - Packet Map of 1997, Arizona Network Intertie Group
Attachment 3 - Packet List of 1999, KF7TP

23 SEPT 91

# Western United States Amateur Packet Radio Network Map

Originated by N7HPR




## LEGEND

- • DIGIPEATER
- ♦ KANODE
- ▪ NETROM
- ▪ THENET
- * HF Gateway PBBS

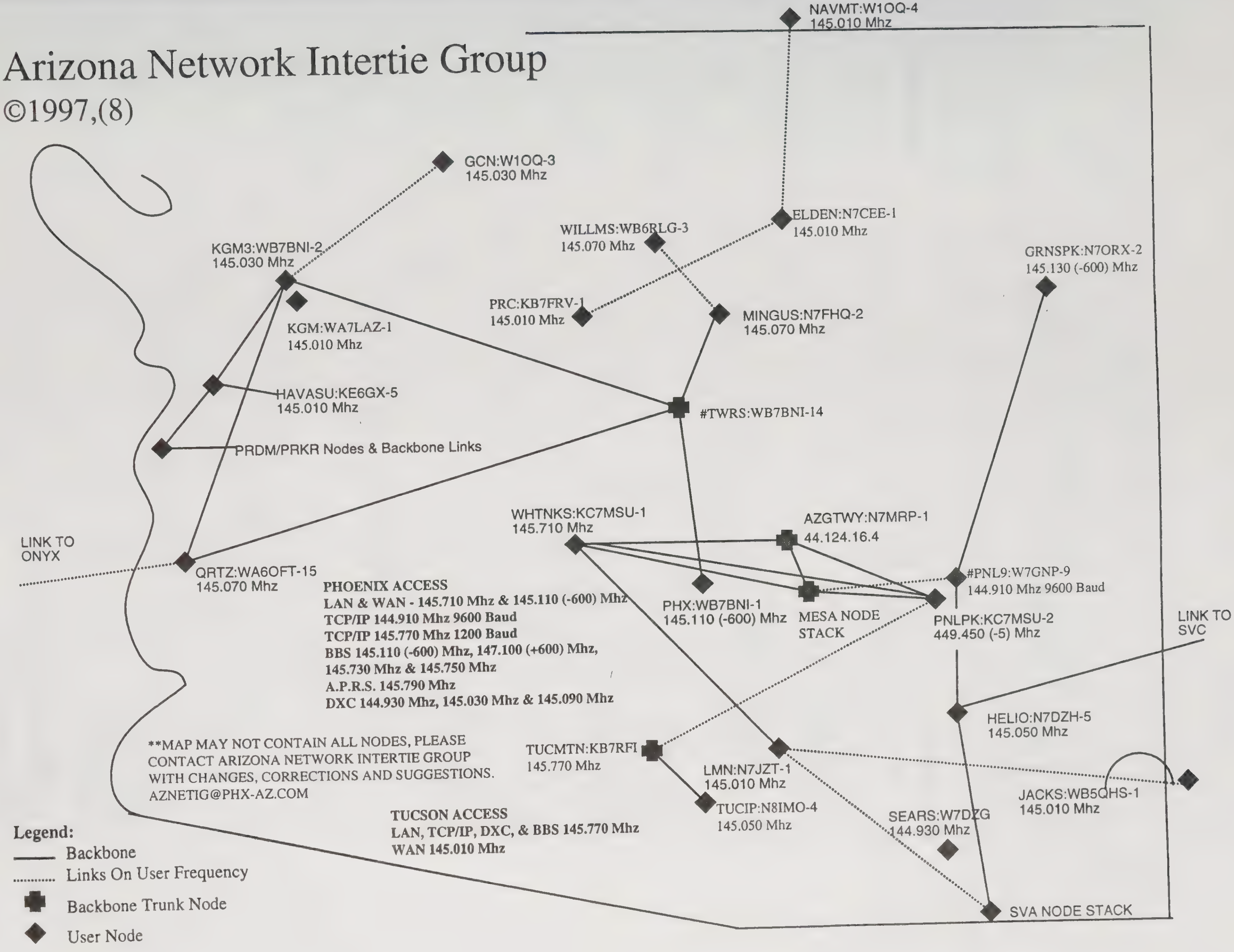
······ Path

— Backbone

FOR LATEST MAP:
SEND UPDATE INFO or AT LEAST TWO ROUTES Lists
& SASE to: **Jim Bradbury : WB5ACL**
**880 Buckhorn Circle**
**Sierra Vista, AZ 85635-1384**
**(WB5ACL @ NJ7P.AZ.USA.NA) >> OFF BISBEE NODE**

# Arizona Network Intertie Group

©1997,(8)



ATTACHMENT 3
ARIZONA PACKET RADIO NODES
by Keith E. Justice, KF7TP

NOTES:
1. Does not include APRS and DX Cluster nodes.
2. Obvious bogus routes were deleted, but no doubt some remain.
3. With the exception of NAVMTN and those shown as out of service, all nodes were accessed by radio from LKSD in August, 1999. Some Internet wormholes were used for intermediate links.
4. Terminal nodes, e.g. ones that do not go anywhere else, are not shown. My apologies to sysops of nodes I missed.
5. Nodes are listed alphabetically within broad geographical regions.

**WHITE MOUNTAINS**

```
Node: GRNSPK:KG7BZ-2  Location: Greens Peak near Springerville
Type: X1JR4 Digital Regenerator Repeater Freq: 145.13 (-600) Speed: 1200
Routes:
  0 HEBER:KC7RAE-6 192 45
  0 LKSD:KF7TP-3 192 43
  0 HTOAD3:N7KQ-13 192 22
  0 WMGATE:KG7BZ-12 192 45

Node: HEBER: KC7RAE-6  Location: Heber
Type: TNOS/LINUX Switch/BBS
Ports: .013 Freq: 145.13 Speed 1200 IP 44.124.34.36
       .010 Freq: 145.01 Speed 1200 IP 44.124.40.4
       .910 Freq: 144.91 Speed 1200 IP 44.124.32.36
Routes :
   Neighbour        Port  PQual Obsocnt Dest
  GRNSPK:KG7BZ-2    .130   192     5     58
  LKSD:KF7TP-3      .130   192     4     50
  HTOAD9:N7KQ-9     .910   200     6     53
  ELDEN:W7MOT-8     .010   192     5     15
  WMGATE:KG7BZ-12   .130   192     6     85
  HTOAD3:N7KQ-13    .130   199     6     35
Comment: Sysop is WA3PNT.  Node uses his wife's call

Node: HTOAD3:N7KQ-13  Location: Clay Springs
Type: X1JR4 Freq: 145.13 Speed: 1200
Routes:
  1 HTOAD9:N7KQ-9 255 36
  0 HEBER:KC7RAE-6 188 33
  0 LKSD:KF7TP-3 188 15
  0 GRNSPK:N7ORX-2 188 22
  0 WMGATE:KG7BZ-12 188 47
Comment: Hardwired to HTOAD9

Node: HTOAD9:N7KQ-9  Location: Clay Springs
Type: X1JR4 Freq: 144.91 Speed: 9600
Routes:
  1 HTOAD3:N7KQ-13 255 28
  0 HEBER:KC7RAE-6 190 53 !
  0 #PNL9:W7GNP-9 190 17
Comment: Hardwired to HTOAD3

Node: LKSD:KF7TP-3  Location: Lakeside
Type: JNOS/DOS Switch Freq: 145.13 Speed: 1200
Ports:  01 Freq: 145.13 Speed 1200 IP 44.124.34.4
       .91 Freq: 144.91 Speed 9600 IP 44.124.32.4
Routes :
   Neighbour           Port  Qual Obs Dest Tries Retries Perc Irtt
   HEBER:KC7RAE-6       13     192  6  108 0      0         0 %
  GRNSPK:KG7BZ-2        13     192  5   54 0      0         0 %
  HTOAD9:N7KQ-9         .91    192  6   89 0      0         0 %
  WMGATE:KG7BZ-12       13     192  6   84 0      0         0 %
   #PNL9:W7GNP-9        .91    192  6   19 34     5        87 %
  HTOAD3:N7KQ-13        13     192  5   81 0      0         0 %

Node: PINEY:N7JVO-2}  Location: Piney Hill, near Fort Defiance
Type: X1JR4 Frequency: 145.01 Speed: 1200
Routes:
  0 GALLUP:W5OXK-1 192 3
```

```
  0 FLV:KJ5KL-2 192 1
  0 ELDEN:W7MOT-8 192 21

Node: WMGATE:KG7BZ-12  Location: Lakeside
Type: TNOS/LINUX Internet Gateway Freq: 145.13 Speed: 1200
Ports:  imo    :  Link to N8IMO  Internet Gateway
        vhf    :  Freq. 145.13 Speed: 1200
Routes :
   Neighbour        Port  PQual Obsocnt Dest
   HEBER:KC7RAE-6   vhf    192     6      7
  GRNSPK:KG7BZ-2    vhf    192     6      5
   TUCIP:KV4OA-4    imo    230     4     74
    LKSD:KF7TP-3    vhf    192     4      7
  HTOAD3:N7KQ-13    vhf    192     3      7
```

**METROPHOENIX**

```
Node: #MESIP:KF7TP-4  Location: Mesa
Type: X1JR4 Frequency: 438.975 Speed: 9600
Routes:
  1 MESVHF:KF7TP-2 255 16 !
  1 KF7TP-5 255 0 !
  1 MESDXC:KF7TP-6 255 1 !
  1 MESTCP:KF7TP 255 39 !
  1 MESPHX:KF7TP-11 255 49 !
  0 #WHTNK:W7MOT-13 192 24

Node: #PHX  Location: Central Phoenix
Type: TheNet Frequency: 6 Meter Backbone Speed: 4800
Routes:
  1 PHX:WB7BNI-1 248 34 !
  0 #TWRS:WB7BNI-14 240 8 !
Comment: Hardwired to PHX

Node: #PNL4:W7GNP-4  Location: Pinal Peak, near Globe
Type: X1JR4 Frequency: 439.350 Speed: 9600
Routes:
  1 #PNL9:W7GNP-9 255 50
Comment: Link to HELIO is not working.

Node: #pnl9:W7GNP-9  Location: Pinal Peak, near Globe
Type: X1JR4 Frequency: 144.91 Speed: 1200
Routes:
  0 HTOAD9:N7KQ-9 192 80
  1 #PNL4:W7GNP-4 255 47
  0 LKSD:KF7TP-3 192 82
  0 TMPE91:W7GNP-5 192 14
  0 MESVHF:KF7TP-2 192 18

Node: #PNLPK:W7MOT-14  Location: Pinal Peak, near Globe
Type: X1JR4 Frequency: 438.975 Speed: 9600
Routes:
  1 PNLPK:W7MOT-7 255 1
  0 #MESIP:KF7TP-4 192 43
  0 #WHTNK:W7MOT-13 192 2
Comment: Hardwired to PNLPK.  Temporarily out of service as of 8/8/99

Node: PNLPK:W7MOT-7  Location: Pinal Peak, near Globe
Type: X1JR4, Digital Regenerator Repeater Frequency: 449.450-600 Speed: 1200
Routes:
  1 #PNLPK:W7MOT-14 255 46
Comment: Hardwired to #PNLPK.  Temporarily out of service as of          8/8/99

Node: #WHTNK:W7MOT-13  Location: White Tanks Mountains, west of Phoenix
Type: X1JR4 Frequency: 438.975 Speed: 9600
Routes:
  1 WHTNKS:W7MOT-6 255 47
  0 #MESIP:KF7TP-4 192 23
  0 AMRC:K7ARC-1 192 1
Comment: Hardwired to WHTNKS

Node: MESDXC:KF7TP-6  Location: Mesa
Type: X1JR4 Frequency: UHF DXC Backbone Speed: 1200
Routes:
```

```
  1 MESVHF:KF7TP-2 255 15
  1 #MESIP:KF7TP-4 255 2
  1 MESPHX:KF7TP-11 255 47
  1 MESTCP:KF7TP 255 32
Comment: Hardwired to Mesa Stack.  Backbone, no direct user connects.

Node: MESPHX:KF7TP-11  Location: Mesa
Type: X1JR4 Frequency: 145.11 Speed 1200
Routes:
  1 MESDXC:KF7TP-6 255 1
  1 #MESIP:KF7TP-4 255 24
  1 MESVHF:KF7TP-2 255 24
  1 MESTCP:KF7TP 255 56
  0 PHX:WB7BNI-1 192 1
Comment: Hardwired to Mesa Stack

Node: MESVHF:KF7TP-2  Location: Mesa
Type: X1JR4 Frequency: 144.91 Speed: 9600
Routes:
  1 MESDXC:KF7TP-6 255 1
  1 #MESIP:KF7TP-4 255 2
  1 MESPHX:KF7TP-11 255 46
  0 TMPE91:W7GNP-5 192 20
  0 #PNL9:W7GNP-9 192 21
  1 MESTCP:KF7TP 255 46
  0 CPHX77:KB7PWD-3 192 17
Comment: Hardwired to the Mesa Stack

Node: PHX  Location: Central Phoenix
Type: TheNet Digital Regenerator Repeater Frequency: 145.11-600
Speed: 1200
Routes:
  1 #PHX:WB7BNI-11 248 7 !
  0 MESPHX:KF7TP-11 192 28 !
Comment: Hardwired to #PHX

Node: TMPE91:W7GNP-5  Location: Tempe
Type: X1JR4 Frequency: 144.91 Speed: 9600
Routes:
  1 TMPE77:W7GNP-3 255 30
  0 #PNL9:W7GNP-9 192 18
  0 MESVHF:KF7TP-2 192 23

Node: TMPE77:W7GNP-3  Location: Tempe
Type: X1JR4 Frequency: 145.77 Speed: 1200
Routes:
  1 TMPE91:W7GNP-5 255 29
Comment: 145.77 is the 1200 baud TCP/IP frequency in Phoenix

Node: W7GNP  Location: North Phoenix
Type: JNOS TCP/IP only, no NetRom
Ports:
.77  Freq: 145.77 Speed 1200 IP: 44.124.4.254
.91  Freq: 144.91 Speed 9600 IP: 44.124.2.254
Comment: Gateway between 1200 and 9600 baud TCP/IP subnets.

Node: WHTNKS:W7MOT-6  Location: White Tanks Mountains west of Phoenix
Type: X1JR4 Frequency: 145.71 Speed: 1200
Routes:
  1 #WHTNK:W7MOT-13 255 47
  0 YUMA:WA3PNT-4 192 36
  0 UNION2:KB7FRV-2 192 1
Comment: Hardwired to #WHNTK
```

**WEST**

```
Node: YUMA:WA3PNT-4  Location: Telegraph Pass east of Yuma
Type: X1JR4 Frequency: 145.71 Speed: 1200
Routes:
  0 WHTNKS:W7MOT-6 192 31
  0 UNION2:KB7FRV-2 0 1 !
  0 YUMAIP:WA3PNT-6 192 54

Node: YUMAIP:WA3PNT-6  Location: Yuma
```

```
Type: TNOS BBS/SWITCH
Ports:
LAN    :  LAN - a local LAN IP 44.124.48.10
.710   Freq: 145.71 Speed 1200 IP 44.124.48.253
.050   Freq: 145.01 Speed 1200 IP 44.124.52.4
.350   Freq: 439.350 Speed 9600 IP 44.124.56.4
.010   Freq: 145.010 Speed 1200 IP 44.124.58.4
Routes :
   Neighbour         Port  PQual Obsocnt Dest
  BLACK:KA6DAC-2    .050   192     6       7
  10XNOD:W7RFI-6    .350   200     6       4
  YUMAIP:WA3PNT-6   LAN      0     6       1
  YUMAIP:WA3PNT-6   .710     0     6      57
  MPK:KA6DAC-1      .050   192     6      18
  YUMA:WA3PNT-4     .710   192     6      17
  UNION:KB7FRV-1    .010   192     6      71
```

**NORTH**

```
Node: #KGM6M:WB7BNI-15  Location: Hayden Peak, near Kingman
Type: TheNet Frequency: 6 Meter Backbone Speed: 4800
Routes:
  1 KGM3:WB7BNI-2 248 1 !
  0 #TWRS:WB7BNI-14 240 2 !
Comment: Hardwired to KGM3

Node: #TWRS  Location: Towers Mtn, near Prescott
Type: TheNet Frequency: 6 Meter Backbone Speed: 4800
Routes:
  0 #KGM6M:WB7BNI-15 240 4 !
  0 #PHX:WB7BNI-11 240 14 !

Node: DAVIS:K7HS-7  Location: Davis Mtn near Prescott
Type: X1JR4 Frequency: 2M DXC Backbone Speed: 1200
Routes:
1   UNION:KB7FRV-1  255  82
Comment: DXCluster backbone, hardwired to UNION

Node: HII03:KB7YKY-2  Location: Lake Havasu City
Type: TheNet Frequency: 145.03 Speed: 1200
Routes:
  0 SNOW03:WY6I-3 192 7
Comment: Provides a route to California nodes.

Node: KB7FRV BBS  Location: Prescott
Type: FBB BBS/Switch with Internet Telnet Gateway
Ports: Freq: 145.01 Speed 1200
       Freq: 145.71 Speed 1200
       Internet Gateway

Node: KGM3  Location: Hayden Peak, near Kingman
Type: TheNet Frequency: 145.03 Speed: 1200
Routes:
  1 #KGM6M:WB7BNI-15 255 6 !
Comment: Hardwired to #KGM6M. In heard list: HII03:KB7YKY-2 Node, has links to Calif.

Node: NAVMTN  Location: Navajo Mountain, east of Page
Type: X1JR4 Frequency: 145.01 Speed: 1200
Comment: This node could not be reached from LKSD

Node PRCVAL  Location: Prescott Valley
Type: BBS/Switch Frequency: 145.01 Speed: 1200
Comment: Temporarily out of service in August, 1999

Node: UNION:KB7FRV-1  Location: Mount Union near Prescott
Type: X1JR4 Frequency: 145.01 Speed: 1200
Routes:
  1 DAVIS:K7HS-7 255 83
  0 YUMAIP:WA3PNT-6 197 17 !
  0 ELDEN:W7MOT-8 192 17
  0 LMN:N7JZT-1 192 13
  0 KGM:WB6RER 192 1
  0 COTTON:N7SBW-1 192 4
  0 FIRE:KC7CHY 192 1
```

```
  0 SCTSDL:KC7AKP-5 192 1
  0 HUMBT:KC7PJO-3 192 1
  0 HEBER:KC7RAE-6 192 61
  0 PINEY:N7JVO-2 192 3
  0 PRCVAL:WD6ETH-2 192 7
```

Node: UNION2:KB7FRV-2 Location: Mount Union, near Prescott
Type: X1JR4 Frequency: 145.71 Speed: 1200
Routes:

```
  0 WHTNKS:W7MOT-6 192 16
  0 YUMA:WA3PNT-4 0 1 !
```

Comment: Links Yavapaie Co EOC with State EOC and Yuma EOC. BBS and Switch via KB7FRV in calls heard but not routes.

**TUCSON**

Node LMN Location: Mount Lemon, north of Tucson
Type: JNOS Frequency: 145.01 Speed: 1200
Routes :

```
  SVA:N7OO-1           1      192   6   37
  SCTSDL:KC7AKP-5      1      140   5    1
  SANMAN:N7CK-3        1      192   6    2
  BISBEE:K7RDG         1      140   3    5
  SONORA:XE2FE-1       1      140   6    1
  JACKS:WB5QHS-1       1      192   6    7
  UNION:KB7FRV-1       1      140   6   14
```

Node: SANMAN:N7CK-3 Location: San Manuel
Type: JOS/LINUX Internet Gateway
Ports:
Available ports:

```
sva    :  AXIP Port to Sierra Vista(AZGATE)
ftl    :  AXIP Port to Ft. Lauderdale, FL
tucson :  AXIP Port to KVOA, Tucson
uhf      Freq: 449.450 Speed 1200 IP 44.124.18.8
01       Freq: 145.01  Speed 1200 IP 44.124.66.32
big      Freq: 145.15  Speed 1200 IP NIL
Routes :
   Neighbour             Port   Qual Obs Dest Tries Retries Perc Irtt
     LMN:N7JZT-1         01      192   6    1 790   140     84 %
   TUCIP:KV4OA-4         tucson 220    6    9 1117  17      98 %
    FTLGW:W4BKX-5        ftl     220   6    1 128   0       100 %
  AZGATE:N7OO-15         sva     220   6   22 283   3       98 %
```

NODE: TUCIP:KV4OA-4 Location: Tucson
Type: TNOS Internet Gateway
Available ports:

```
wm     :  -> AXIP Port to White Mountains, AZ (c wmgate)
simi   :  -> AXIP Port to Simi Valley, CA (c simi smivly)
segate :  -> AXIP Port to Sierra Vista, AZ (c segate)
sva    :  -> AXIP Port to Sierra Vista, AZ (c azgate)
lcr    :  -> AXIP Port to Las Cruces, NM (c nmsugw)
sanman :  -> AXIP Port to San Manuel, AZ (c sanman)
nwla   :  -> AXIP Port to Northwest Louisiana (c ipcpw)
reno   :  -> AXIP Port to Reno, NV (c wadg)
delrio :  -> AXIP Port to Del Rio, TX (c riogw)
phx    :  -> AXIP Port to Phoenix
2m     :  -> AX25 Radio port, TCP/IP LAN (145.050)
223    :  -> Mail backbone port, Tucson
Routes :
   Neighbour             Port   Qual Obs Dest Tries Retries Perc Irtt
  GBNODE:W5GB-8    (BPQ) lcr     200   6    4 2421  185     92 % 1074266232
  TUCSON:WB7TLS-1  (BPQ) 223     230   5    2 2202  317     87 %
  TUCSON:WB7TLS-1  (BPQ) 2m      192   5    2 4     64       5 %
  SEGATE:W7DZG-4   (BPQ) segate 200    5    3 339   14      96 %
  SANMAN:N7CK-3          sanman 220    6    4 220   2       99 %
  7C4008:W6SWE-5         2m      192   6    1 0     0        0 %
  WMGATE:KG7BZ-12        wm      230   6    2 364   29      92 %
  AZGATE:N7OO-15         sva     230   6   67 286   22      92 %
```

Node TUCSON:WB7TLS-1 Location: Tucson
Type: BPQ BBS/Switch
Ports:

```
  1 Freq: 145.05 Speed 1200 IP: NIL
```

```
  2 Freq: 220MHz Backbone Speed 9600, IP: NIL
Routes:
  2 KV4OA-4 10 3!
  1 KA7TXS-2 10 1
  1 N7MDT-1 100 6
  1 N7OO-2 100 32
  2 W7DZG 255 6
  1 KV4OA-4 10 1!
  1 K7EAR-5 10 2
  1 W6SWE-5 10 1
```

**SOUTHEAST**

```
Node:#HELIO:K7EAR-4  Location: Heliograph Peak west of Safford
Type: X1JR4 Frequency: 439.350 Speed: 9600
Routes:
  1 HELIO:K7EAR-5 255 83
  0 SEARS:W7DZG 192 8
  0 2HEADS:KA7TXS-4 192 8
Comment: Hardwired to HELIO

Node: 2HEADS:KA7TXS-4  Location: Dos Cabezas Peak, south of Safford
Type: Unknown Switch
Ports:
Port 1: Freq: 144.930 Speed 1200 IP ?
Port 2: Freq: 439.350 Speed 9600 IP ?
Routes:
  2 W7DZG 192 7

Node: AZGATE:N7OO-15  Location: Sierra Vista
Type: TNOS Internet Gateway
Routes :
   Neighbour        Port  PQual Obsocnt Dest
   KSOLA:NW0I-6      ks     230     6      9
  WAGATE:N7NEI-8     wa     230     6     14
  NIHRAC:K3YGG-2     md     230     5      1
   TUCIP:KV4OA-4     tus    230     6      4
  SANMAN:N7CK-3      sml    230     5      1
  OREGON:WB7AWL-10   or     230     5     19
   HFXNS:VE1SMU-3    ns     230     5     13
   UTDXC:NG7M-8      utdx   230     4     19
   ALWGW:KB5CDX-8    wal    230     4     16
  IPCAPE:W0PLW-2     mo     230     6      1
    HARC:VE3THA      ve3    230     6      8
   LINUX:VE3MCH-10   ont    230     5      4
   QMNGW:IW0QMN-10   it     230     5      5
   BENCA:WH6IO-7     ca     230     5      6
   AZLAN:N7OO-6      ax0    230     5      4
Available ports:
fla    :  Port to MIAMI:AE4EJ-1 in Miami, FL
tus    :  Port to TUCIP:KV4OA-4 in Tucson, AZ
la     :  Port to #TAXLA:WB0TAX-14 in Elm Grove, LA
or     :  Port to OREGON:WB7AWL-10 in Talent, OR
mo     :  Port to IPCAPE:W0PLW-2 in Cape Girardeau, MO
md     :  Port to NIHRAC:K3YGG-2 in Bethesda, MD
wa     :  Port to WAGATE:N7NEI-8 in Seattle, WA
ont    :  Port to LINUX:VE3MCH-10 in Hamilton, Ontario
wal    :  Port to ALWGW:KB5CDX-8 in Walla Walla, WA
ns     :  Port to HFXNS:VE1SMU-3 in Halifax, Nova Scotia
ks     :  Port to KSOLA:NW0I-6 in Olathe, KS
it     :  Port to QMNGW:IW0QMN-10 in Perugia, Italy
sml    :  Port to SANMAN:N7CK-3 in San Manuel, AZ
dx     :  Type 'dx' for the N7OO DXCluster (Western US area)
utdx   :  Port to NG7M-8 in Clearfield, UT
ut     :  Port to UUGATE:WA7SLG-3 in Salt Lake City, UT
ca     :  Port to BENCA:WH6IO-7 in San Francisco, CA
ve3    :  Port to HARC:VE3THA in Halton, Ont
ax1
ax0    :  Link to 145.01

Node: AZLAN:N7OO-6  Location: Herford
Type: X1JR4 Frequency: 145.01 Speed: 1200
Routes:
  1 SVA5:N7OO-2 248 19
```

```
  1 SVA:N7OO-1 248 91
  0 AZGATE:N7OO-15 230 58
Comment: Hardwired to SVA5 and SVA

Node: AZSON:N7MDT-4  Location: Gold Hill, near Nogales
Type: BBS/Switch Frequency: 145.01 Speed: 1200
Routes:
  0 N7MDT-1 255 5
  1 N7JZT-1 192 5!
  1 XE2FE-1 192 1!
Comment: Hardwired to NOGAZ

Node: BISBEE:K7RDG  Location: Mule Mtn, near Bisbee
Frequency 145.010 Speed: 1200
Comment: Hardwired to SEARS.  Apparently out of service in August, 1999.

Node:HELIO:K7EAR-5  Location: Heliograph Peak west of Safford
Type: X1JR4 Frequency: 145.05 Speed: 1200
Routes:
  0 NOG:KA7TXS-2 192 68
  1 #HELIO:K7EAR-4 255 83
  0 SVA5:N7OO-2 192 52
  0 NOGAZ:N7MDT-1 192 5

Node: NOG:KA7TXS-2  Location: Red Mtn near Nogales
Type: TheNet Frequency: 145.05 Speed: 1200 Speed: 1200
Routes:
  0 HELIO:K7EAR-5 192 58
  0 TUCIP:KV4OA-4 192 57
  0 CRC:W7SA-5 192 52
  0 7C4008:W6SWE-5 192 1
  0 NOGAZ:N7MDT-1 192 4
  0 TUCSON:WB7TLS-1 192 2
Comment: Solar Powered

Node: NOGAZ:N7MDT-1  Location: Gold Hill near Nogales
Type: BBS/Switch Frequency: 145.05 Speed: 1200
Routes:
  0 N7MDT-4 255 5
  1 W7SA-5 192 5
  1 KA7TXS-2 192 5
  1 KV4OA-4 192 0
  1 W6SWE-5 192 0
Comment: Hardwired to AZSON

Node SEARS:W7DZG  Location: Mule Mtn neaqr Bisbee
Type: JNOS Switch, BBS
Ports:
UHF   Freq: 438.925  Speed 9600
BBONE Freq: 439.350  Speed 9600
VHF   Freq: 144.930  Speed 1200
220   Freq: 220MHz   Speed 9600
Routes :
   Neighbour             Port  Qual Obs Dest Tries Retries Perc Irtt
  2HEADS:KA7TXS-4        VHF    144   6    1 0     0        0 %
  2HEADS:KA7TXS-4        bbone  192   6    1 2     2       50 %
     DX1:N7BXX-1         VHF    192   5    1 0     0        0 %
  TUCSON:WB7TLS-1        220    128   2    2 2221  738     75 %  TBOSS:KK7RV           UHF
192   6    2 0     0        0 %
  SEGATE:W7DZG-4   (BPQ) UHF    192   6   47 125   9       93 % 15

Node: SEGATE:W7DZG-15  Location: Sierra Vista, Arizona, USA DM41VM
Type: TNOS Internet Gateway
Available ports:
fyrom  :  -> AXIP Port to N1NGN Former Yugoslav Republic Of Macedonia
nh     :  -> AXIP Port to AE1T- in Plymouth New Hampshire
boston :  -> AXIP Port to KB1BWD in Boston MA  :  -> AXIP Port to HG5BDU in Budapest,
Hungary
tucip  :  -> AXIP Port to N8IMO in Tucson, AZ
slo    :  -> AXIP Port to S55TCP in Slovenia, Europe
sanman :  -> AXIP Port to N7CK in San Manuel, AZ
zg     :  -> AXIP Port to 9A0TCP in Zagreb, Croatia
nwla   :  -> AXIP Port to KB4CPW in Northwest Louisiana
zurich :  -> AXIP Port to HB9AE in Zurich, Switzerland
italy  :  -> AXIP Port to IW0QMN in Perugia, Italy
```

```
uhf    :  -> AX25 Radio port, TCP/IP LAN (438.925)
vhf    :  -> Unused 1200 BPS PORT
Routes :
   Neighbour              Port   Qual Obs Dest Tries Retries Perc Irtt
   LJUTCP:S55TCP-6        slo     220   6   18 474    45      91 %
   TBOSS:KK7RV            uhf     220   6    1 25     8       75 %
   IPCPW:KB4CPW-8         nwla    220   6    5 10     0       100 %
   TUCIP:KV4OA-4          tucip   220   6    7 33     0       100 %
   ZHGATE:HB9AE           zurich  220   6   41 277    24      92 %
   ZGTCP:9A0TCP           zg      220   6   12 111    25      81 %
   QMNGW:IW0QMN-10        italy   220   6    5 23     2       92 %
   SEARS:W7DZG            uhf     200   6    2 104    7       93 %

Node: SIGBOX:N1NGN  Locatoin: Southeast Arizona
Type: NOS BBS/SWITCH
Ports:
Freq: 144.930 Speed 1200
Freq: 145.010 Speed 1200
Freq: 145.050 Speed 1200
Freq: 438.925 Speed 9600
Freq: 14.105 (USB)Speed 300
Comment: Apparently off the air in August, 1999

Node: SVA:N7OO-1  Location: Near Sierra Vista
Type: X1JR4 Frequency: 145.01 Speed: 1200
Routes:
  0 LMN:N7JZT-1 243 43 !
  1 SVA5:N7OO-2 248 20
  1 AZLAN:N7OO-6 248 99
Comment: Hardwired to SVA% and AZLAN

Node: SVA5:N7OO-2  Location: Hereford
Type: X1JR4 Frequency: 145.05 Speed: 1200
Routes:
  1 SVA:N7OO-1 240 98 !
  1 N7OO-4 240 0 !
  1 AZLAN:N7OO-6 240 97 !
Comment: Hardwired to AZLAN and SVA

Node: TBOSS:KB7ZGA  Location: Sierra Vista
Type: JNOS Switch
Available ports:
hf
930
440   Freq: 438.925 Speed: 9600
2m    Freq: 145.010 Speed: 1200
Routes:
   Neighbour              Port  Qual Obs
   SEGATE:W7DZG-4   (BPQ) 440    192   4
   SEARS:W7DZG            440    192   5
```

*A Software Implementation for Federal Standard 1052 (Mil. Std. 188-110A HF Modems)*

Robert McGwier, N4HY
64 Brooktree Road
East Windsor, NJ 08520-2438
Email: rwmcgwier@home.com

## Abstract

Federal Standard 1052 is a modem designed for use on HF. It is specifically designed to overcome the effects of propagation on HF to a certain degree. This paper describes the approach taken by the author in a software implementation for the personal computer. The author has chosen to do the initial implementation on Pentium and Dec-Alpha based computers running Linux.

Keywords: Serial HF Modem Federal Standard 1052 Linux PC

## Introduction

In the late 1980's, the author was one of several people who pushed digital processing and its myriad applications and possibilities (McGwier, et. al.1-3). During the early years, our personal computers did not have sufficient power to do the myriad tasks one needed to do in order to implement an extremely sophisticated modem, such as Mil. Std.188-110A. We were users of digital signal processing chips, and while these DSP chips have gotten faster, our general purpose computer microchips have sky rocketed in their power to accomplish difficult tasks.

During the 1990's, we have seen the publication and distribution of many software packages implementing increasingly complex signal processing software for a variety of applications, including a demodulator for the Advanced Composition Explorer (Karn, 4), an HF demodulator designed by Peter Martinez (Jacob, 5), and Forward Error Correction software (Karn, 6). The quality of the sound equipment in computers has steadily improved, as has the drivers needed to efficiently utilize them. The ubiquitous Soundblaster™ by Creative, Inc. has enabled much to be done with our computers.

Almost from the outset of our DSP efforts, there has always been the desire to produce a really high quality HF modem for use by amateur radio operators. It has also always been clear that it was going to take a really serious effort by a signal processing professionals to implement this in the short term. There have been several commercial modems, many of them really good designs. We will not list them here for fear of leaving one or more out. These being proprietary designs, none of them have been as good a learning experience as they could have been. Since no serious signal processing professional has come forward, and the long term has arrived, we will demonstrate an implementation of the Mil. Std. 188-110-A1 demodulator. It is written entirely in C, since I can't force myself to learn C++. It uses standard sound cards for the PC to deliver digital samples to the processor from your audio source and to render audio for your radio. In this paper we will give a cursory overview of the design of the demodulator. After receiving all of the necessary approvals from my employer, the source code will be released under a Gnu Public License. It is the hope of the author that this will increase serious experimentation on HF in a way that we have seen fostered by the release of the latest software design by Martinez (6). Most of this paper will be about the actual standard with the exception of the highlights from the demodulator. Another paper, will be done elsewhere with all of the gory details.

## Federal Standard 1052 (formerly Military Standard 188-110-A1)

During the 1980's, the US adopted a standard design for communications using the HF. Its design considerations were primarily to allow more reliable communications on HF that were more reliable than standard FSK based RTTY signals and to make use of the rapidly advancing digital signal processing chips. A block diagram of the design is a little complicated to follow but is included for completeness.

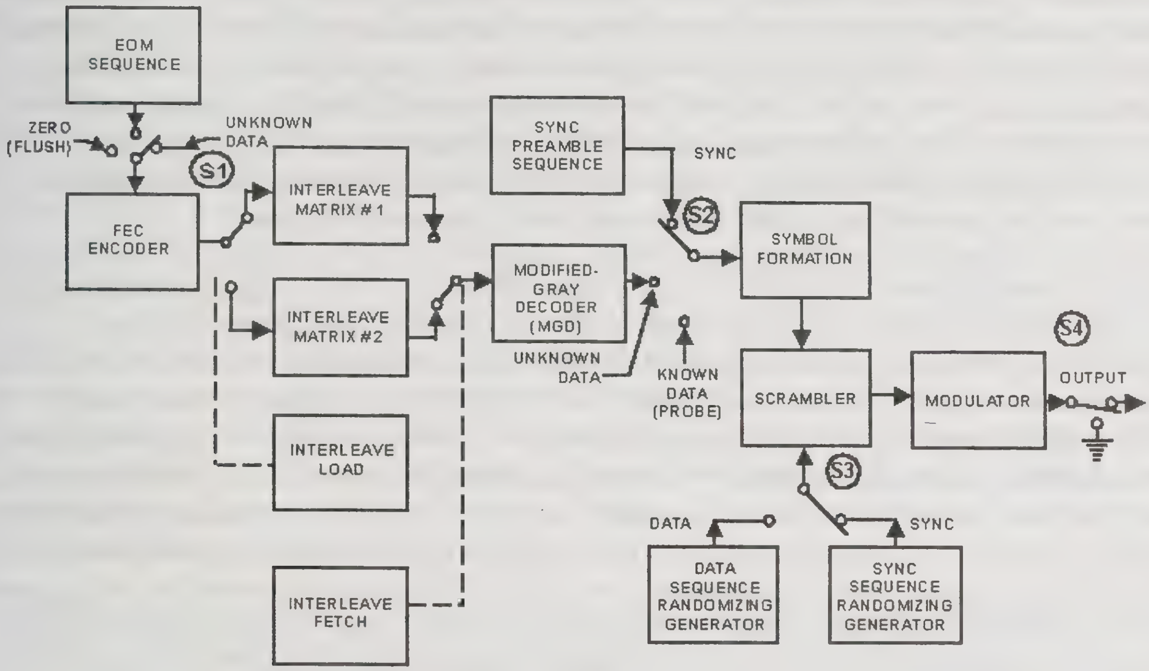



**Figure 1. A block diagram of the serial modems operations**

The design adopted an 8 PSK signal at 2400 baud as bearer of information symbols. Since it is 8 PSK, every baud carries three bits of data. Data is encoded by changing the phase by one of eight values: 0, pi/4, pi/2, 3pi/4, and so on, up to 7 pi/8. Since there are eight different angles we could change the phase by, this allows us to transmit three bits of information. So theoretically, we could be sending 7200 bits per second of data with this signal. This would not work very well on HF to say the least.

The first thing that we would find out is that we have no idea where in the data we are, and if we did, which of the eight possible rotations of the data could be the correct one for our communications purpose. That is, we would not know which of eight possible phases to use to decode the data. In order to overcome this, we will transmit training data. This training data in the HF serial modem world and elsewhere is called a preamble. We will send the following numbers, remembering that we send three bits at a time. Thus we will transmit numbers between 0-7. When we are sending SYNC, the switch S2 in figure one is in the SYNC position.

The synchronization pattern to transmit is

0, 1, 3, 0, 1, 3, 1, 2, 0, D1, D2, C1, C2, C3, 0.

The three-bit values of D1 and D2 are very important to our communications channel. They choose at what speed we will transmit data and simultaneously, how much protection we will enforce to help out our data. When we are attempting to find a signal like this, we look for the pattern 0,1,3,0,1,3,1,2,0 in the data being received and then assume we are getting a valid sync and then listen carefully to the rest of the preamble. The rest of the preamble consists of several copies of everything but C1,C2, and C3 being repeated. C1, C2, and C3 are counters and count down to zero in a special way. After we get to zero, we know that we are to immediately begin to decode what follows as the actual message. Why do we do this? This allows us several chances at detecting that the signal is present. Once we do detect, we are given several pieces of data that are known to us. When you are transmitting known data, it is much easier to measure what effects the HF channel is having on your data, what frequency offset exists between you and the transmitting station, and finally to determine the timing needed to synchronize the receiver and the transmitter. This is the hardest aspect of all such demodulators to get right. Harris RF though their way of doing the channel mitigation job was so nice that the patented it. We do things differently here and is the real impetus for doing this work. We'll return to that later.

While it is straightforward to understand 10,9,8....1,0 blast off for the C1, C2, and C3, the purpose of D1, and D2 require more explanation. It is quite straightforward to understand one aspect of the function of D1 and D2. The worse the channel conditions are, clearly, the slower we must transmit data in order to be successful in getting our message across. But wait, I already said that we were always going to send data at 2400 baud. How can we reconcile these two different views? In order to transmit data at a lower effective rate, even though we are transmitting at 2400 baud, we will use redundancy to help us conquer the noise and interference on the channel. An easy to understand (but dumb) kind of redundancy comes when you just repeat what you are saying and voting on what is the most likely message given the many copies. Here the redundancy comes from forward error correction and in some cases additional redundancy from multiple copies, but we ***never*** majority vote. The first layer of redundancy is done by a standard IBS rate ½ constraint length 7 convolutional code. See Figure 2.

Phil Karn, KA9Q, has done a very nice decoder for this exact code(6). We have gladly stolen that code under the GPL. Convolutional codes are extremely powerful tools for providing for noisy channels in the unlikely (ever) case we have additive Gaussian noise. This is not a good model for HF noise. The noise has a whitish component, but it is very impulsive. This causes burst errors. Convolutional codes do not handle burst errors well in general. A clever device has been introduced to alleviate the fact that the noise is highly correlated. One way would be to decorrelate the effects of the noise by spreading its effects out in time. This is extremely effective and much of the new hot topic in coding, Turbo codes can attribute their success in large part due to the very large decorrelators. These are called interleavers. In Fed. Std. 1052, you take a block of data and apply the FEC to it. You then take the encoded bits and permute their order in time. This has been studied in depth by mathematicians and engineers as to how to do this "optimally." Such a pemutation is used here. You then transmit the encoded+permuted data. If burst errors occur now, their effects are spread out on the receiving end by applying the inverse of the permutation or deinterleaver. This does a great job at decorrelating the noise. The faster you send data, the more time you need to spread the bits out so larger permutations are used for the faster data. Each interleaver takes exactly the same amount of time, irrespective of how fast the data is transmitted. On good channels given a data rate, one uses a short interleaver. Given the same data rate, as the channel degrades, one can go to a long interleaver to help. The disadvantage of the interleaver is the group delay through it. Short interleaver are 0.6 seconds long and long interleavers are 4.8 seconds long!

And still, this is not enough. The channel is just too erratic or *nonstationary* for the estimate we made about the channel during the preamble to be true or effective enough to help us for long. For this reason, ***probes*** are sent interspersed amongst the encoded and interleaved data. These probes are known data. Everyone has agreed ahead of time (in the standard) what the probes will be and when they will be sent. This allows you to relearn the channel to some degree even in the midst of sending unknown data (the message) to the receiver. This allows you to adapt much more quickly to known data in the channel and to remove frequency offset, timing drift, as well as other channel effects. How much known data is transmitted with the unknown data is a function of the D1, D2 values. Table 7 gives the values for the "channel symbols" used for D1 and D2. Table 8 gives the actual bit values to these channel symbols. The thing to keep in mind when trying to understand why things are done this way, it is to allow for a lousy channel where possible. D1,D2 are streams of bauds. One using correlation against all of the known values for D1,D2 in order to decide what is the most likely actual values of D1 and D2 when you are receiving them. In most of the cases amateurs will use, there will be 20 probes sent for every 20 data bauds. This will give a pretty reliable 1200 bps. Only in the case of 2400 bps do you use 32 unknown data bauds to 16 probes. Data rate, 2400 bps, is not used frequently by the commercial and government users of this standard.

75 bps is handled very differently and is a miserable thing to code but worth every bit of the pain involved in getting it right. There are no known data probes sent. Long sequences of data are sent for each bit and cross correlation is used to determine which values are being sent. Since these have such tight correlation peaks, they are very good data in and of themselves for learning about the channel and attempting to produce good data . This mode will copy data that simply cannot be heard by the ear on an interference free open band channel.

For Phil's decoder to work, soft symbols (probabilities) have to be computed for each bit. One more signal processing trick has made the demodulator better than it otherwise would have been. As you can see from Figure 2., the data is not sent out in the order 0,1,... as we move around the points on the circle for the constellation values. In all cases but one, any two constellation values decode to a pattern of bits that differ in at most one bit. Intuitively, this is done because the most likely error in a noisy environment, full of clutter, is to mistake one neighbor for another. This is helped tremendously by giving neighbors to the extent possible, very similar addresses. Let's take an example. Suppose I compute a soft symbol to be at 75 degrees with a magnitude of 0.75 and I am transmitting at 1200 bps so I am using the (XY) or dibit symbols. Notice that I am above and to the right of the line from 135 degrees, through the origin, down to 315 degrees. Notice that both symbols here have the same first bit. Thus when I compute a probability to send to Phil's decoder, I am going to be pretty certain that the first bit is a zero. I will be less certain about the second bit but I will claim it looks more like a one than a zero because it is closer to 90 degree line. This type of division of the circle in ways that allow you compute probabilities where some of the bits are really certain is called modified-Gray decoders. Given the way we compute symbols, the probabilities fall right out.

Some miscellaneous information is that the following scrambling sequence for the sync preamble shall repeat every 32 transmitted symbols:

7 4 3 0 5 1 5 0 2 2 1 1 5 7 4 3 5 0 2 6 2 1 6 2 0 0 5 0 5 2 6 6

This is added symbol by symbol to the phase of the transmitted data just before the preamble is sent out on the air. In addition to this a randomizer is added to the data symbols and the probes (all zeros) just before they are transmitted. This is a 480 long bit pattern, which is 160, 3 bits symbols, which are added just before transmission. They are derived from the shift in Figure 3 with initial fill 101110101101.

## The Software Implementation

We will concentrate here on what is new in the demodulator since everything else is just implementation details. In all demodulators for serial modems on HF, something must be done to overcome the effects of the channel on the data. The performance requirements placed in Table 10 show that you must overcome channel defects that are at times severe. In addition to this, we would also like to be more robust in the face of in band, narrow interferers, where we have chosen to put an adaptive notch filter. All demodulators to date, known to the author, attempt to solve the following problem. Suppose we are giving a stream of data samples coming from an A/D. Let's call this stream Y(t), where t = the integers and correspond to sample times. Let X(t) be zero stuffed data, that is we will assume that the Y(t) are being issued at a rate that is an integer multiple of the baud rate. We will put zeroes in the X(t) where it is not time for a data baud. We assume the following model for the observed data:

$$Y(t) = \mathrm{Sum}_{i=1}{}^{L} h(l)X(t-l) + W(t).$$

Here h(l) are the filter coefficients which we assume will model the channel operating on the data and W(t) is noise and we assume it is additive.

It is typical to assume (whether or not it is true) that the filter h is approximately invertible. A filter is then applied to the Y(t) that produces a soft symbol. Sometimes there is added to this feedback of the previous decisions X^(t). An adaption procedure is implemented to minimize the error between the new soft symbol or block of symbols and the hard decisions for X. In some cases, no feedback is done at all. In this case, not all of the intersymbol interference is removed that it is possible to remove. If Harris uses in their radios, exactly what they have patented, then they are not doing decision feedback in the traditional sense but are working their way in from the probes on either side of the unknown data.

Very bad things happen to the equalizer approach and other inversion approaches when the filter defined by h has zeroes in its spectrum. All sorts of tricks have to played to fix the numerical instabilities induced. Trying to invert the channel through an equalizer where the channel has a bunch of zeroes in its spectrum is the mathematical equivalent of dividing by zero.

A better approach would be to take the point of view that we wish to guess putative data symbols, and then find the best filter given this data estimate that will produce the observations Y(t). Given this new filter estimate, one can then find better data. This sounds like an iterative procedure. It is. The first step, given a filter find the data, is called Expectation. The next step of re-estimation of the filter coefficients is called Maximization and the algorithm which implements it is commonly called the EM algorithm as a result. On its face, it is too expensive. If one knew the channel response, *a priori,* one could do the viterbi algorithm and this would indeed outperform the equalizer methods normally chosen but it is very expensive. There is also no theoretically clean way to derive better filter coefficients in an iterative fashion from the viterbi algorithm output other than least squares given the output data. This

coupled with the cost of producing the data makes it prohibitive. Lets take a different approach. For completeness we give the following very dry mathematical rendering of what we do here approximately. It is taken directly from the original paper by Dempster, et. al. (8). If you study the model we have given for the A/D samples, Y(t), it is clear that the data values X(t) are hidden from our view by the filter h, and the additive noise W(t). Since X are assumed to be independent in a certain way, they have nice properties and are a type of mathematical stochastic process called a Markov Chain. This is an extremely powerful concept and allows the analysis and algorithm which follow.

**The EM Algorithm in general**

Let X be the set of hidden variables, Y the set of observable variables, and Omega_X and Omega_Y the set of values or sample space for these variables. The product of Omega_X and Omega_Y can be thought of as the space of complete samples and Omega_Y the set of incomplete samples. If (x,y) is a complete sample, then y is the observable part.

Let f(x, y | theta) specify a family of functions one of which governs the generation of complete samples and let g(y | theta) specify a family of functions one of which governs the generation of incomplete samples. The functions f and g are related by the following equation.

```
g(y | theta) = Integral_{x in Omega_X} f(x,y | theta) dx
```

The EM algorithm attempts to find a ``value for theta which maximizes g(y | theta) given an observed y, but it does so by making essential use of the associated family f(x, y | theta)'' [Dempster et al., 1977 ](7).

EM can be thought of as a deterministic version of Gibbs sampling. EM operates on the means or modes of unknown variables using expected sufficient statistics instead of sampling unknown variables as does Gibbs sampling. Both EM and Gibbs sampling are used for approximation with incomplete data.

Suppose that x is the missing data and y is the observed data. It may be straightforward to calculate p(x, y | theta) but not so easy to calculate p(y | theta).

Define Q(theta | theta') as follows:

```
Q(theta | theta') = E[log p(x, y | theta) | theta', y]
```

where x is the random variable and we are taking the expectation with respect to the probability density p(x | theta',y).

The EM algorithm works as follows

1. Set i to 0 and choose theta_i arbitrarily.
2. Compute Q(theta | theta_i)
3. Choose theta_i+1 to maximize Q(theta | theta_i)
4. If theta_i != theta_i+1, then set i to i+1 and return to Step 2.

where Step 2 is often referred to as the expectation step and Step 3 is called the maximization step.

The generalized EM (GEM) algorithm is the same except that instead of requiring maximization in Step 3 it only requires that the estimate be improved.

Here is a proof sketch showing that each iteration of EM actually improves the current estimate theta.

We start with the following simple identity (see the Appendix at the end of this document):

```
log p(y | theta) = log p(x, y, | theta) - log p(x | theta, y)
```

and take expectations of both sides treating x as a random variable with distribution p(x | theta',y).

```
log p(y | theta) = E[log p(x, y | theta) | theta', y] -
                   E[log p(x | theta, y) | theta', y]
(1)
```

We note without proof that the second term in (1) is minimized when theta = theta' (see Exercise 1 below).

Now consider any value theta'' such that

```
E[log p(x, y | theta'') | theta', y] >
E[log p(x, y | theta' ) | theta', y]
```

and note that if we replace theta' by theta'' we increase the first term in (1) while not increasing the second term so that

```
p(y | theta'') > p(y | theta')
```

thereby improving our current estimate.

In general, computations required for EM can be quite difficult requiring complex integrations to compute the required expectations and perform the maximization. In the remainder of this section, we consider a special case of EM well suited to distributions in the exponential family.

Consider the problem of learning a model for the unsupervised learning problem in which X and Y are boolean variables, Y depends on X, and X is hidden. In this case, complete samples (x,y) are drawn from {0,1}^2, incomplete samples are drawn from {0,1}.

The hypothesis space is defined by the following network structure.

```
o ----> o
X       Y
```

Here are the parameters for the above structure.

```
{theta(x) = Pr(X=x) : x in {0,1}}

{phi(x,y) = Pr(Y=y | X=x) : (x,y) in {0,1}^2}
```

Here is a graphical model representing the learning problem.

```
            ____________________
            |                  |
            |                  |
o ----> o ----> o <---- o          The boxed part of the
theta(X) |   X       Y   | phi(X,Y)   model indicates a plate
            | N                |      (see [Buntine, 1994])
            |__________________|      representing N samples.
```

Here are some samples with missing values indicated by underscores.

```
X       Y
---------
_       1        Note that in the general case the
_       0        values for X could be either partially
_       1        or completely missing in the data.
_       0
_       0
   ...
```

We represent the parameter distributions using beta distributions. Let's assume that Beta[n,m] is the distribution that corresponds to out having observed n 0's and m 1's. For each parameter distribution, the associated counts correspond to sufficient statistics. In the case of the theta parameters, we have

```
        n(x) = Sum_{s in S} (1_{Value(s,X) = x})
```

where S is the set of samples comprising the data, s is a variable ranging over samples, Value(s,V) is the value assigned to the variable V in sample s, 1_{Value(s,V) = v} = 1 if Value(s,V) = v, 1_{Value(s,V) = v} = 0 if Value(s,V) != v. In the case in which there is no missing data, we have

```
        xi(theta(1) | S) = Beta(n(0),n(1))
```

Similarly, for the phi parameters, we have

```
        n(x,y) = Sum_{s in S} (1_{Value(s,Y) = y} 1_{Value(s,X) = x})
```

and, again in the case of no missing data, we have

```
        xi(phi(1,1) | S) = Beta(n(1,0),n(1,1))
```

In the case of missing data, instead of using the sufficient statistics which we don't have, we use the expected sufficient statistics to iteratively assign estimates to theta and phi. This iterative process proceeds in two steps.

In the first step, called the expectation step, we compute the expected sufficient statistics as follows.

```
    E(n(x) | S,theta,phi) =
        Sum_{s in S} Pr(X = x | Y = Value(s,Y), theta, phi)

    E(n(x,y) | S,theta,phi) =
   Sum_{s in S} (Pr(X = x|Y = Value(s,Y),theta,phi)1_{Value(s,Y) = y})
```

The required computations can easily be carried out for distributions in the exponential family.

In the second step, called the maximization step, we set theta and phi to their mode conditioned on the expected sufficient statistics. In this case, finding the mode is trivial. For example, if the prior on phi(1,1) is Beta[alpha_1,alpha_2], then we're looking for the mode of

```
     Beta[alpha_1+E(n(1,0)|S,theta,phi),alpha_2+E(n(1,1)|S,theta,phi)]
```

which is just

```
                           alpha_2+E(n(1,1)|S,theta,phi)
 phi(1,1)' = -------------------------------------------------------
------
                           [alpha_1+E(n(1,0)|S,theta,phi) +
              alpha_2+E(n(1,1)|S,theta,phi)]
```

The formulation of EM given here is perfectly suited to working with distributions in the exponential families, including the Bernoulli, Poisson, normal, gamma, beta, multinomial, and Dirichlet distributions.

## Our Implementation

Chrystomos Nikias and Minn Shao (9) gave nice implementation in a demodulation problem which is almost like ours. They assumed real data (ours is complex), they assumed all unknown data, we make use of the fact that our data is surrounded by known data in all the modes except 75 bps. There, the problem is more like theirs but we still have several things going for us. They make several simplifying assumptions that do cost you performance but allow the algorithm to be adapted to real time processing. Rather than assume that all moments of the distributions are necessary above, they assume that only first and second moments and pairwise conditional expectations are needed in a clever way.

This simple but powerful assumption allows one to approximate the total likelihood problem given above in a meaningful and implementable way. The viterbi algorithm mentioned above, given N observations, L length filter, and M symbols in our constellation costs $O(NM^L)$ operations and $O(M^L)$ memory. As you can see, it pays to allow filter length's L, which are inadequate in order to gain the power of the dynamic program. This is prohibitively expensive. While the full EM algorithm greatly reduces the memory requirement, the cost is still $O(NM^L)$. Under the assumptions in (9), which we have experimentally tested, this cost drops to $O(NM^2)+ O(LN^2)$ operations and $O(N^2)$ memory. This is completely feasible and is similar in cost to the Harris patented equalization algorithm while in no way resembling. It uses approximate total likelihood instead and reestimates filters and data together an algorithm derived from appropriate use of Bayes rule and a Markov model of the data.

## Results so far

The demodulator works on the air. We will demonstrate it at the conference. The 75 bps version should outperform all known amateur radio modems with a cost. The occupied bandwidth is a full SSB bandwidth but greatly reduced powers are required for adequate communications.

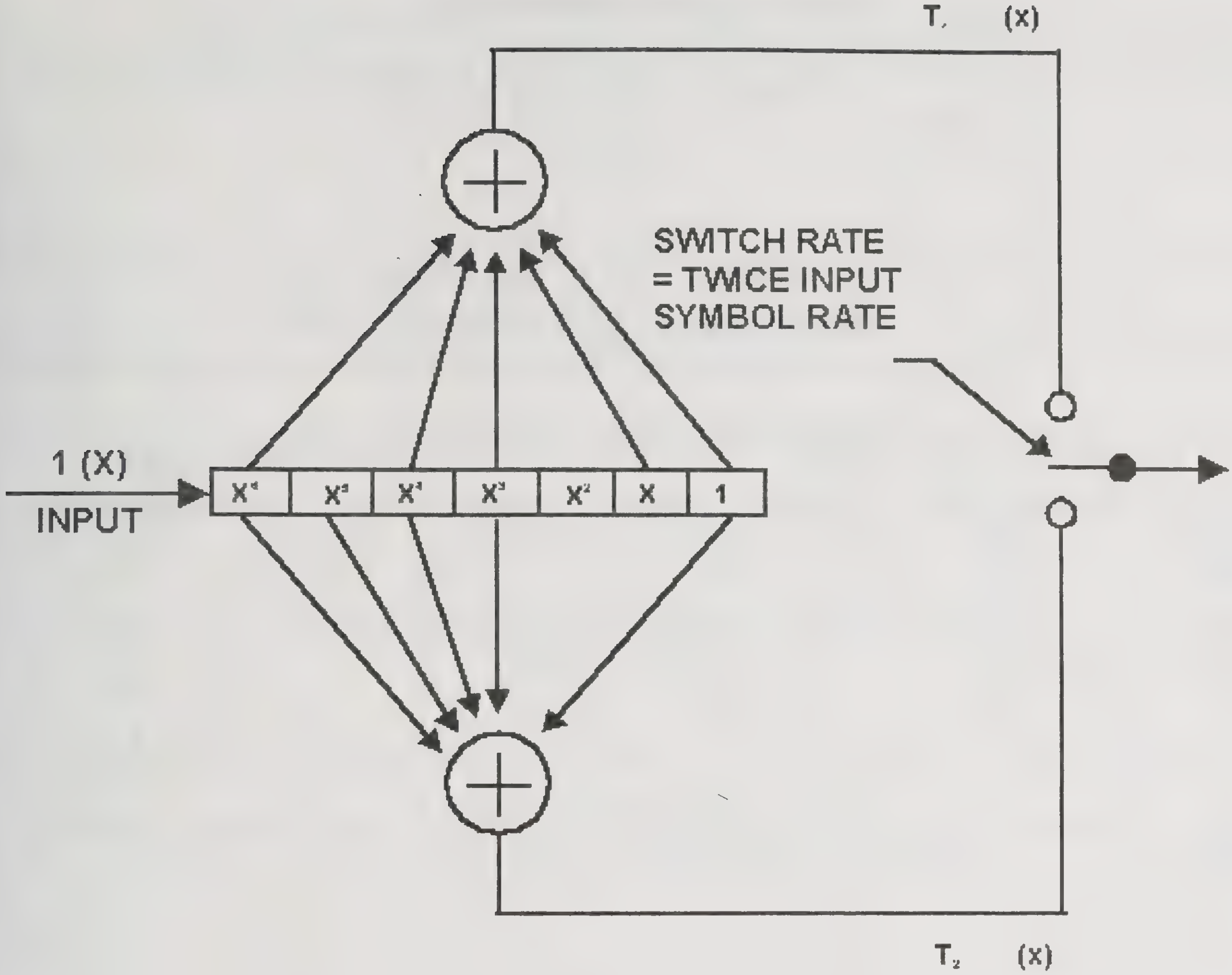



**Figure 1. Rate ½ K=7 convolutional code**
$T_1(x)=x^6+x^4+x^3+x+1$
$T_2(x)=x^6+x^5+x^4+x^3+1$

**Table 1. Error-correcting coding**

| DATA RATE (b/s) | EFFECTIVE CODE RATE | METHOD FOR ACHIEVING THE CODE RATE |
|---|---|---|
| 4800 | (no coding) | (no coding) |
| 2400 | 1/2 | Rate 1/2 |
| 1200 | 1/2 | Rate 1/2 code |
| 600 | 1/2 | Rate 1/2 code |
| 300 | 1/4 | Rate 1/2 code, repeated 2 times |
| 150 | 1/8 | Rate 1/2 code, repeated 4 times |
| 75 | 1/2 | Rate 1/2 |

**Table 2. Interleaver matrix dimensions**

| | LONG INTERLEAVER | | SHORT INTERLEAVER | |
|---|---|---|---|---|
| Bit rate (b/s) | Number of rows | Number of columns | Number of rows | Number of columns |
| 2400 | 40 | 576 | 40 | 72 |
| 1200 | 40 | 288 | 40 | 36 |
| 600 | 40 | 144 | 40 | 18 |
| 300 | 40 | 144 | 40 | 18 |
| 150 | 40 | 144 | 40 | 18 |
| 75 | 20 | 36 | 10 | 9 |

**Table 3. Bits-per-channel symbol**

| Data rate (b/s) | Number of bits fetched per channel symbol |
|---|---|
| 2400 | 3 |
| 1200 | 2 |
| 600 | 1 |
| 300 | 1 |
| 150 | 1 |
| 75 | 2 |

**Table 4. Modified-Gray decoding at 4800 b/s and 2400 b/s**

| INPUT BITS | | | |
|---|---|---|---|
| First bit | Middle bit | Last bit | Modified-Gray decoded value |
| 0 | 0 | 0 | 000 |
| 0 | 0 | 1 | 001 |
| 0 | 1 | 0 | 011 |
| 0 | 1 | 1 | 010 |
| 1 | 0 | 0 | 111 |
| 1 | 0 | 1 | 110 |
| 1 | 1 | 0 | 100 |
| 1 | 1 | 1 | 101 |

**Table 5. Modified-Gray decoding at 1200 b/s and 75 b/s**

| INPUT BITS | | |
|---|---|---|
| First bit | Last bit | Modified-Gray decoded value |
| 0 | 0 | 00 |
| 0 | 1 | 01 |
| 1 | 0 | 11 |
| 1 | 1 | 10 |

**Table 6. Bits-per-channel symbol**

| Data rate (b/s) | Number of bits fetched per channel symbol |
|---|---|
| 2400 | 3 |
| 1200 | 2 |
| 600 | 1 |
| 300 | 1 |
| 150 | 1 |
| 75 | 2 |

**Table 7. Assignment of designation symbols D1 and D2**

| | SHORT INTERLEAVE | | LONG INTERLEAVE | |
|---|---|---|---|---|
| BIT RATE | D1 | D2 | D1 | D2 |
| 4800 | 7 | 6 | - | - |
| 2400 (secure voice) | 7 | 7 | - | - |
| 2400 (data) | 6 | 4 | 4 | 4 |
| 1200 | 6 | 5 | 4 | 5 |
| 600 | 6 | 6 | 4 | 6 |
| 300 | 6 | 7 | 4 | 7 |
| 150 | 7 | 4 | 5 | 4 |
| 75 | 7 | 5 | 5 | 5 |

**Table 8. Channel symbol mapping for sync preamble**

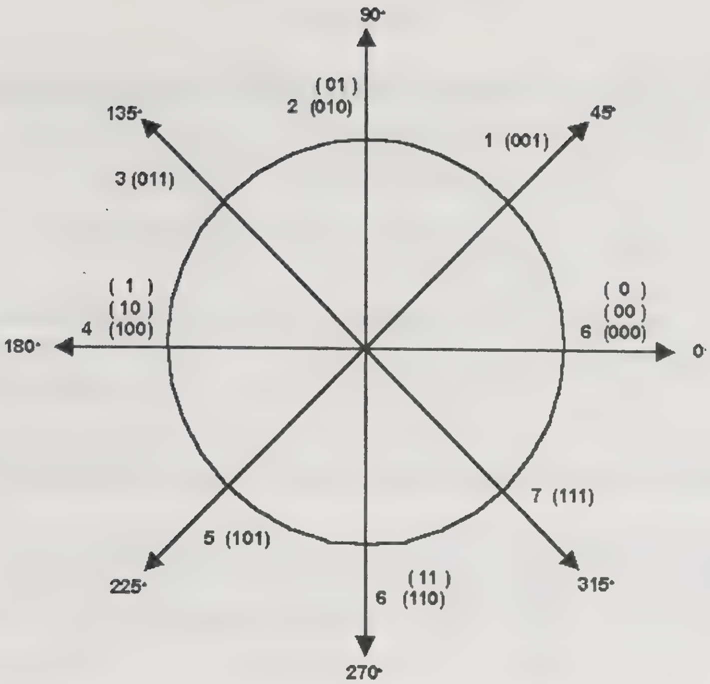
| CHANNEL SYMBOL | TRIBIT NUMBERS |
|---|---|
| 000 | (0000 0000) repeated 4 times |
| 001 | (0404 0404) repeated 4 times |
| 010 | (0044 0044) repeated 4 times |
| 011 | (0440 0440) repeated 4 times |
| 100 | (0000 4444) repeated 4 times |
| 101 | (0404 4040) repeated 4 times |
| 110 | (0044 4400) repeated 4 times |
| 111 | (0440 4004) repeated 4 times |

**Table 9. Data phase waveform characteristics**

| Information rate | Coding rate | Channel rate | Bits/channel symbol | 8-phase symbols/ channel symbol | No. of unknown 8-phase symbols | No. of known 8-phase symbols |
|---|---|---|---|---|---|---|
| 4800 | (no coding) | 4800 | 3 | 1 | 32 | 16 |
| 2400 | 1/2 | 4800 | 3 | 1 | 32 | 16 |
| 1200 | 1/2 | 2400 | 2 | 1 | 20 | 20 |
| 600 | 1/2 | 1200 | 1 | 1 | 20 | 20 |
| 300 | 1/4 | 1200 | 1 | 1 | 20 | 20 |
| 150 | 1/8 | 1200 | 1 | 1 | 20 | 20 |
| 75 | 1/2 | 150 | 2 | 32 | all | 0 |

**Table 10. Fed Std. Serial (single-tone) mode minimum performance.**

| User bit rate | Channel paths | Multipath (ms) | Fading BW (Hz) [1] | SNR (dB) [2] | Coded BER |
|---|---|---|---|---|---|
| 4800 | 1 Fixed | - | - | 17 | $1.0 \times 10^{-3}$ |
| 4800 | 2 Fading | 2 | 0.5 | 27 | $1.0 \times 10^{-3}$ |
| 2400 | 1 Fixed | - | - | 10 | $1.0 \times 10^{-5}$ |
| 2400 | 2 Fading | 2 | 1 | 18 | $1.0 \times 10^{-5}$ |
| 2400 | 2 Fading | 2 | 5 | 30 | $1.0 \times 10^{-3}$ |
| 2400 | 2 Fading | 5 | 1 | 30 | $1.0 \times 10^{-5}$ |
| 1200 | 2 Fading | 2 | 1 | 11 | $1.0 \times 10^{-5}$ |
| 600 | 2 Fading | 2 | 1 | 7 | $1.0 \times 10^{-5}$ |
| 300 | 2 Fading | 5 | 5 | 7 | $1.0 \times 10^{-5}$ |
| 150 | 2 Fading | 5 | 5 | 5 | $1.0 \times 10^{-5}$ |
| 75 | 2 Fading | 5 | 5 | 2 | $1.0 \times 10^{-5}$ |




LEGEND:

0°...315° = PHASE (DEGREES)

0...7 = TRIBIT NUMBERS

(000)...(111) = THREE-BIT CHANNEL SYMBOLS

(00)...(11) = TWO-BIT CHANNEL SYMBOLS

(0)...(1) = ONE-BIT CHANNEL SYMBOLS

**Figure 2. Symbol Values**


NOTES:

1. INITIAL SETTING SHOWN
2. SHIFTED 8 TIMES BETWEEN OUTPUTS

**Figure 3. Linear Feedback Shift Register for scrambling**

## Reference


(1) *Digital Signal Processing and Amateur Radio*, Thomas A. Clark, W3IWI, and Robert W. McGwier, N4HY Sixth Computer Networking Conference Proceedings, 1987.

(2) *DSP Modems: It's Only Software,* Robert W. McGwier, N4HY, Sixth Computer Networking Conference Proceedings, 1987.

(3) *The DSP Project Update*, by Thomas Clark, W3IWI and Robert McGwier, N4HY, Seventh Computer Networking Conference, 1988.

(4) *Ace Demodulator,* **http://people.qualcomm.com/karn/code/acedemod**, Phil Karn, KA9Q

(5) *PSK31 Modem*, **http://aintel.bi.ehu.es/psk31.html**, Eduardo Jacob, EA2BAJ

(6) *Forward Error Correction Codes*, **http://people.qualcomm.com/karn/code/fec/**, Phil Karn, KA9Q

(7) *Federal Standard 1052*, **http://ntia.its.bldrdoc.gov/~bing/fs-1052/htm** (formerly MS-188-110A1).

(8) "Maximum Likelihood from incomplete data via the EM algorithm", A.P. Dempster, N.M. Laird, D.B. Rubin, Journal of the Royal Statistical Society, Vol. B-39, pp. 1-37, 1977

(9) "An ML/MMSE Estimation Approach to Blind Equalization", Shao, M., Nikias, C., IEEE Transactions on Communications, IV, 1994, ISBN 0-7803-1775-0/94, pp. 569-572.


## Biography


Robert McGwier is N4HY. He has published numerous articles in ham radio journals and participated in many AMSAT and TAPR projects. He wrote the demodulators in the AEA DSP2232 and Quiktrak tracking software for AMSAT. He participate in the Microsat project and was a member of the board of directors for AMSAT. While having been mostly absent from ham radio for the last 8 years, he has concentrated on family, Boy Scouts, Golf, and work. Bob resides with his wife, Shann, N2HPE and three children at 64 Brooktree Road, East Windsor, NJ. He is rwmcgwier@home.com.

# Next Generation of Amateur Radio Systems

Paul L. Rinaldo, W4RI
4559 Quality Street
Fairfax, VA 22030 USA
E-mail: prinaldo@arrl.org

The following PowerPoint presentation was originally prepared for a small group of representatives of ARRL, DARC and UBA at Friedrichshafen, Germany. During the discussion, it was acknowledged that HF digital voice was difficult (using LPC-10 or MELP, for example) and that early emphasis should be placed on VHF/UHF multi-media systems. This slide set is offered for your information.

**Next-Generation**
**Amateur Radio Systems**

◊

*Renewal of Amateur Radio*
*using new technologies*

Doomsaying

- Amateur Radio has peaked
- Memberships are declining
- Equipment sales are falling
- Attendance at events is reducing
- PCs and the Internet are more interesting
- Commercial services want our spectrum
- The noise level is rising

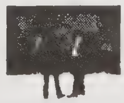
12 August, 1999 NGEN 4

What if we do nothing?

- We will lose licensees, members
- Manufacturers will abandon market
- Regulators will reallocate our spectrum
- Bands above 146 MHz could be lost
- Amateur Radio becomes more CB-like

12 August, 1999 NGEN 5

We are in this together



- The problem and solution are *global*
- Most developments involve *teamwork*
- Individual contributions not just technical
- Amateurs and industry must *co-operate*
- IPR *advice* is needed

12 August, 1999 NGEN 6

## ARRL initiatives

- 1997: Created 3 technical awards:
  - Service, innovation, microwave development
- 1998: White paper at Porlamar
- 1999: President's initiative
  - Technology Task Force
  - Technology Working Group

12 August, 1999 NGEN 7

## Challenges

- Retain our spectrum
- Fulfill spectrum requirements
- Revitalise HF before sunspots drop
- Develop SHF and EHF systems
- Integrate satellite and terrestrial systems

12 August, 1999 NGEN 8

## Entering a new environment

- Knowledge-based society/information age
- Regulatory environment should permit innovation, digital modes
- Greater competition for spectrum
- Increasing noise pollution/power lines
- Automotive environment evolving

12 August, 1999 NGEN 9


## Automobiles migrating to 42 volts

- Power-hungry options excessive for 12 V
- MIT consortium developing 42-V standard
- Transition may include dual voltages
- Amateur radio equipment could benefit
- Possible need for adaptive power supplies

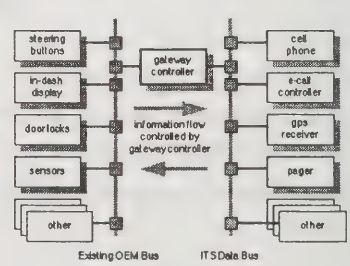
12 August, 1999 NGEN 10

## SAE ITS data bus

- Electronic devices installed in cars are several generations old as automotive design takes 4 times as long.
- IDB to facilitate *plug-and-play*.
- Cell phones, GPS, roadside ITS services
- Should amateurs take account of IDB?

12 August, 1999 NGEN 11

## SAE IDB architecture




12 August, 1999 NGEN 12

## Impact of Internet

- Using Internet as *worm hole*
- Remote station operation
- Preventing commercial use of amateur radio
- Embedded microprocessors
- What degree of integration is appropriate?
- Internet 2?

12 August, 1999 NGEN 13

## Other services are already digital

- Cellular enters 3G, studying 4G
- Digital dispatch land mobile
- Digital sound and TV broadcasting
- IP telephony becoming telephone standard
- Multimedia (audio/video/data)
- Satellites

12 August, 1999 NGEN 14

## Amateur Radio systems will be digital

- Improve end-to-end quality of service
- Take advantage of new technologies
- Provide multimedia (audio/video/data)
- Operate multi-mode (terrestrial/satellite)

12 August, 1999 NGEN 15

## Enabling technologies

- Error detection and correction (ARQ, FEC)
- Standards to evaluate:
  - Coding: MPEG, JPEG, H.261, H.323...
  - Digital dispatch: TETRA, APCO 25…
  - Public: IP, 2G, SMS, 3G, 4G, FWA
  - Digital sound BC: IBOC, DRM...
  - Digital TV: HDTV, interactive…
  - Digital cameras, recorders

12 August, 1999 NGEN 16

## Radio technologies

- Access methods (FDMA, CDMA, TDMA)
- Spectrum-efficient modulation methods
- Software defined radios (SDRs)
- Automatic link establishment (ALE)
- Adaptive power control (APC)
- Global location: GPS, GLONASS, APRS

12 August, 1999 NGEN 17

## Framework to innovate is needed

- Ability to introduce new protocols
- Download software from Web
- Need to recognise protocol in use
- No regulatory restrictions on digital modulation

12 August, 1999 NGEN 18

## Tower of Babel

- Many protocols in use: CW, SSB, RTTY, AX.25, PACTOR, G-TOR, CLOVER, PSK31, SSTV, TV…

*Possible approaches to selection*

- Option 1: Group by band plan
- Option 2: Signal recognition
- Option 3: Link establishment protocol

12 August, 1999 NGEN 19

## Link-establishment protocol

***Essential features***

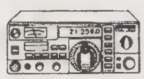
- Ubiquitous
- Identifies stations
- Establishes contact
- Negotiates protocols
- Robust waveform

***Optional features:***

- Authentication
- Give location
- Adaptive (ALE)
- SMS messaging
- Scalable rate

12 August, 1999 NGEN 20

## HF digital radios



- SSB is 50 years old.

***HF digital voice could have these features:***

- Improved voice quality
- Fading and interference reduction
- Greater spectrum efficiency
- Facilitate ≥2.4 kbit/s data, images

12 August, 1999 NGEN 21

## Digital repeaters

1 0 1 0 11...

- >56 kbit/s
- Simplest digital repeater
- Multimedia repeaters
- Web server type
- Digital fast-scan TV
- Complexity at repeater
- Simplicity at outstations

12 August, 1999 NGEN 22

## Amateur satellites

AMSAT

- Satellites should be integral part of next-generation amateur systems
- Develop VHF/UHF LEO clusters to operate with hand-held earth stations
- Promote satellites for bands ≥24 GHz

12 August, 1999 NGEN 23

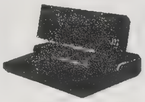
## Facilitating hardware design

- Experimenters lose time and motivation
- Test bed is needed
  - Built-in power supply
  - Bus for common power and signals
  - Over top for IF/RF interconnection
  - Board under test accessible

12 August, 1999 NGEN 24

## Investigate available hardware

- Personal computers



-  Cellular phones
- Digital signal processing boards
- Integrated circuits, their use and misuse

12 August, 1999 NGEN 25

## Antenna development

- Restricted space antennas
- Adaptive (smart) antennas
- Broadband antennas
- Diversity antennas (and receivers)

12 August, 1999 NGEN 26

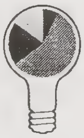
## Applications

*The next generation should include:*

- a global disaster communication capability
- human resources development opportunity
- proof-of-performance experiments, such as rural communications in developing countries
- Propagation research

12 August, 1999 NGEN 27

## Intellectual property rights



***An IPR policy is needed that will:***

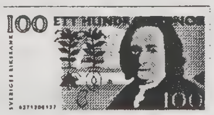
- protect individual rights of inventors
- make new products available
- ensure fair and equitable treatment
- help provide a stable manufacturing environment

*We must benefit from past mistakes and avoid new disputes*

12 August, 1999 NGEN 28

## Financing the next generation

- Mostly self-funded by those involved
- National societies
- Technical clubs
- Grants




12 August, 1999 NGEN 29

## Publication of information

- National society journals
- Web site for next generation
  - Open, closed or both?
  - Languages: English, French, Spanish, German, Italian, Japanese, Russian, Arabic?
- ARRL willing to disseminate tech info

12 August, 1999 NGEN 30

# An Inexpensive High Speed Modem for the Universal Serial Bus (USB)

Thomas Sailer, HB9JNX/AE4WA,
and
Johannes Kneip, DG3RBU

August 3, 1999

**Abstract**

This article describes a simple and inexpensive modem intended to link end users at 76.8kBit/s to the high speed backbone network. The modem can be connected to standard PC's using the Universal Serial Bus (USB).

## 1 Introduction

Last year, we presented a modem design for the Enhanced Parallel Port (EPP). Three years ago, when the basic design decisions for that project were made, the Enhanced Parallel Port was the only ubiquitous high speed interface available on PC's and Laptops.

In the mean time, a new high speed interface emerged: the Universal Serial Bus (USB). The new interface offers several advantages over the Enhanced Parallel Port:

- high transmission speed, 12MBit/s
- serial transmission, i.e. thinner and cheaper cables
- easier port sharing, up to 127 devices per port by using hubs
- rigorous specification

Furthermore, the necessary infrastructure like operating system drivers for popular operating systems like Windows98, Linux, FreeBSD and NetBSD is either in place or emerging at a rapid pace, and USB accessories like hubs can be bought at every computer store.

Therefore, we decided to develop a modem similar to the EPP design, but with an USB interface.

## 2 Design Goals

Our goal was to provide a flexible design that supports a wide variety of applications without sacrificing cost effectiveness.

Our design supports the following modes of operation:

1. FSK (G3RUH compatible) from 9.6 kBit/s up to about 300 kBit/s
2. 1200 Baud AFSK
3. External Modem using standard Modem Disconnect interface [4]
4. Audio IO

1 covers the same applications as the EPP modem designs [7, 8]. 2 makes this design attractive for 1200 Baud users by offering a smooth upgrade path to higher speeds. 3 allows it to accomodate modems for nonstandard modulation formats, and 4 together with suitable application software offers the same features as a standard FM transceiver, but software controllable.

If the Modem Disconnect port is not used to connect to a modem, it may be used as eight general purpose PIO pins.

The circuit also has a port to control the operation of a T7F [10] transceiver. The T7F is a popular synthesizer 9600 Baud data radio design. An RSSI ADC makes the signal strength accessible to the PC, four digital lines allow the channel selection, one digital line selects 12.5kHz or 25kHz channel spacing, and a 1200 Baud asynchronous UART allows the channel memory to be programmed.

## 3 Design Considerations

Figure 1 shows the prototype circuit diagram. There will likely be changes for the production unit, so do not expect the final drivers to work with that circuit.

In order to shorten the development cycle, we decided to base the new design on the successful EPP design [7, 8]. On first sight, the circuit diagrams of both modems look very similar, but there are some important differences, as explained below.

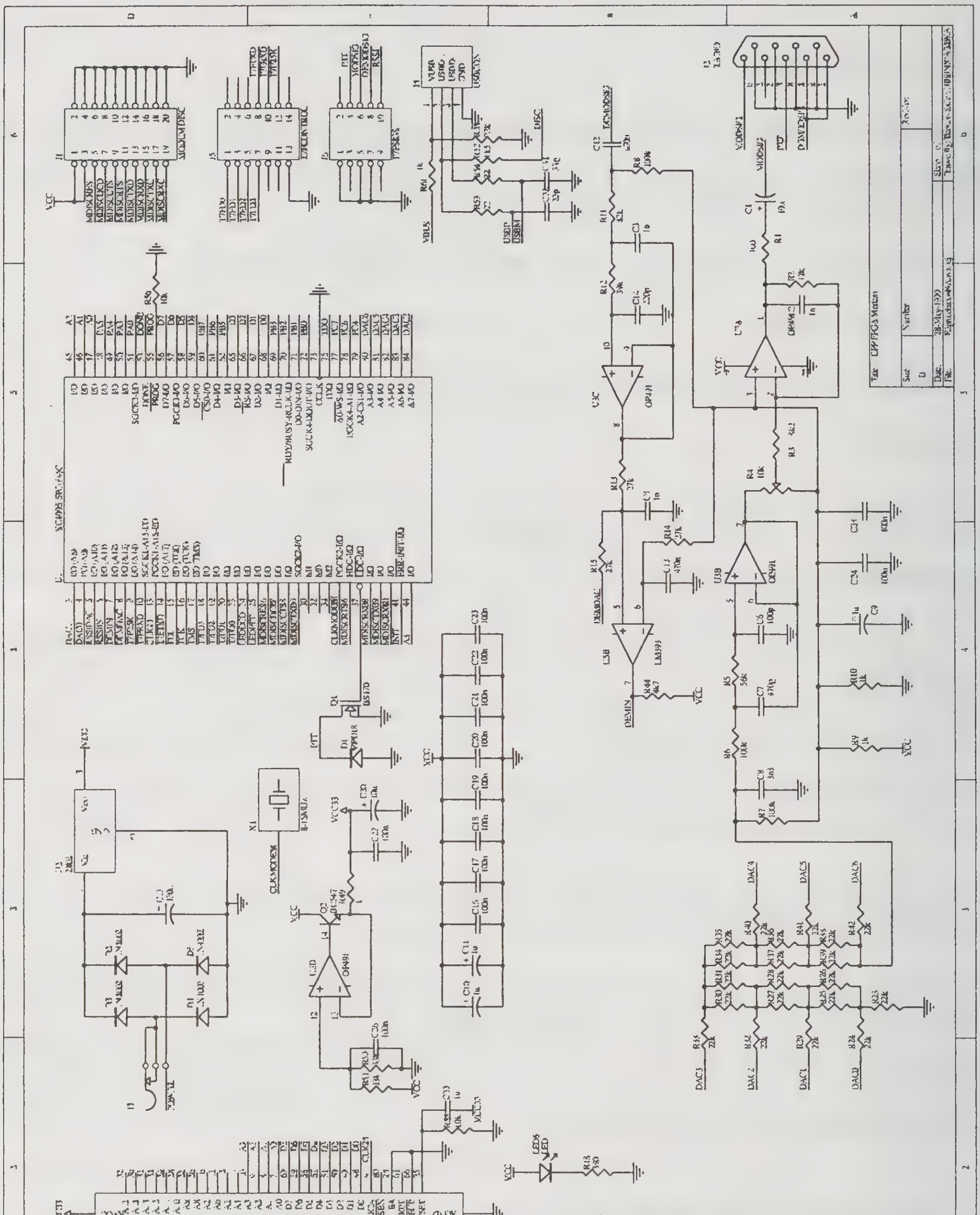
### 3.1 Microcontroller selection

The USB bus specification requires the devices to be able to respond to relatively complicated inquiries. This necessitates a microcontroller on the device and makes the implementation of an USB controller on a FPGA[1] infeasible. In April 1999, we therefore surveyed the market for suitable microcontrollers with built-in full speed (12 MBit/s) USB interface engines (table 1).

The AMD 186CC looks ideal for our purpose on first sight. It is however quite expensive, and since it doesn't contain any memory on chip, requires external FLASH and SRAM memories, making it even

[1] "Field Programmable Gate Array": programmable logic devices [1]

Figure 1: Prototype Circuit Diagram – see text



| Manufacturer | Device | CPU core | features | problems |
|---|---|---|---|---|
| AMD | Am186CC | 8086 | HDLC | $ |
| AnchorChips | AN2131 | 8051 | 8kB SRAM, no nonvolatile memory needed | slow microcontroller core |
| Cypress | 7C64213, 7C64313 | proprietary | 8kB EPROM 256B SRAM, 1kB FIFO SRAM, master DMA into FIFO SRAM | not yet available, simple microcontroller |
| Infineon | SAB-C541U | 8051 | 8kB ROM, 256B SRAM | very slow microcontroller core, not enough SRAM |
| Motorola | MPC823 | PowerPC | HDLC, ethernet | $$, BGA package |

Table 1: April 1999 market survey for microcontrollers with USB interface engine

more expensive. It looks however promising for a small node application with three radio channels and one USB channel. We already have a prototype design for that.

The AnchorChips part has 8k of on-chip SRAM. Its distinguishing feature however is that the complete firmware for the CPU can be downloaded from the USB bus. It doesn't require any nonvolatile program storage. Its disadvantage however is the relatively slow CPU core. It's faster than the original 8051, but access to external data and USB data is inefficient in the 8051 architecture. But overall the AnchorChips part is still a good choice for our purpose.

Unlike the EPP modem, the microcontroller already has on chip SRAM and the USB bus is designed for low latencies, an additional SRAM for buffering is not necessary.

## 3.2 Power Supply Considerations

USB delivers 5V over the standard four conductor wire. However, the specification allows huge tolerances to accomodate the resistive loss in the cabling which makes it impossible to directly power 5V devices without a DC/DC converter. It is however possible to feed a low drop 3.3V regulator.

Most USB controllers are 3.3V devices, the AnchorChips part including. It features 5V tolerant IO pins, too.

Since powering the analog circuitry with only 3.3V would result in output voltage swings too low for many transceivers in use, we decided to power the analog part with 5V, and to simplify the interface of the analog part to the FPGA we used a 5V FPGA. In fact, the analog circuitry is the same as in the EPP modem design [7, 8].

The USB interface may supply up to 500mA[2], which is approximately 2.5 Watts. Typical 9600 Bauds data radios have around 7 Watts of HF output power, therefore requiring about twice the supply power. So it is impossible to feed the radio from the USB port, and therefore at least the radio requires an external

[2] only if the device is connected to a self powered hub

power supply, typically 12V DC. Given that it does not make sense to use an expensive DC/DC converter to power the modem from the USB bus, so we decided to require an external supply for the modem too.

## 3.3 Firmware Download

In order to achieve the flexibility of the design outlined in the design goals section, the modem design employs two programmable devices, namely the Field Programmable Gate Array (FPGA) and the USB microcontroller. Apart from a small $I^2C$-EEPROM there is no nonvolatile storage in the modem. The EEPROM only stores USB vendor and product ID codes to make the design distinguishable from other AnchorChips designs.

To make the modem operational, the firmware for both programmable IC's needs to be downloaded from the host PC. The first step for the host PC is to download special code for the microcontroller whose only purpose it is to accept data from the USB bus and download it to the FPGA. Once the FPGA is configured, the host PC stops the microcontroller again and downloads the firmware for the particular operating mode requested.

The microcontroller then simulates the unplugging and replugging of the modem from the USB bus. This forces the operating system to reread the USB interface tables from the modem again. This allows the modem to have different USB requirements than the default ones provided by AnchorChips, and it also allows the modem to supply another product ID to force the operating system to load another driver.

## 3.4 The FSK Mode

Unlike the EPP modem design which encodes and decodes HDLC in the driver running on the host PC, the USB modem exchanges raw packets with the PC. Therefore, the modem has to implement the HDLC encoding and decoding process.

HDLC processing can be classified into two categories: Bit oriented tasks like (un)stuffing, CRC calculation and flag detection, and memory management oriented tasks, like storing a frame until the CRC is received and then determining whether to keep it or not. It turns out that the former can be done very easily on the FPGA, while FPGA's are quite unsuited to the latter.

The microcontroller communicates with the FPGA mainly through a register interface connected to its address and data bus. The interface consists of 16 8Bit registers aliasing over the whole external part of the XDATA address space of the CPU.

Small FIFOs decouple the HDLC encoder/decoder and the microcontroller. The FIFOs store eight bits of data and a one or two bit tag word indicating if the data is to be considered as "payload" data or "meta" data.

The transmit FIFO may be accessed at four addresses, which determines the two tag bits. The tag bits are listed in table 2.

In the receive case, the tag is only one bit wide and distinguishes between payload and meta data bytes. A meta data byte is inserted when a HDLC flag or abort is received. The data byte then contains indications whether a flag or abort was received, and in the former case whether the CRC was correct or not and the number of residual bits. Since processing a HDLC flag is an expensive operation (the old packet has to

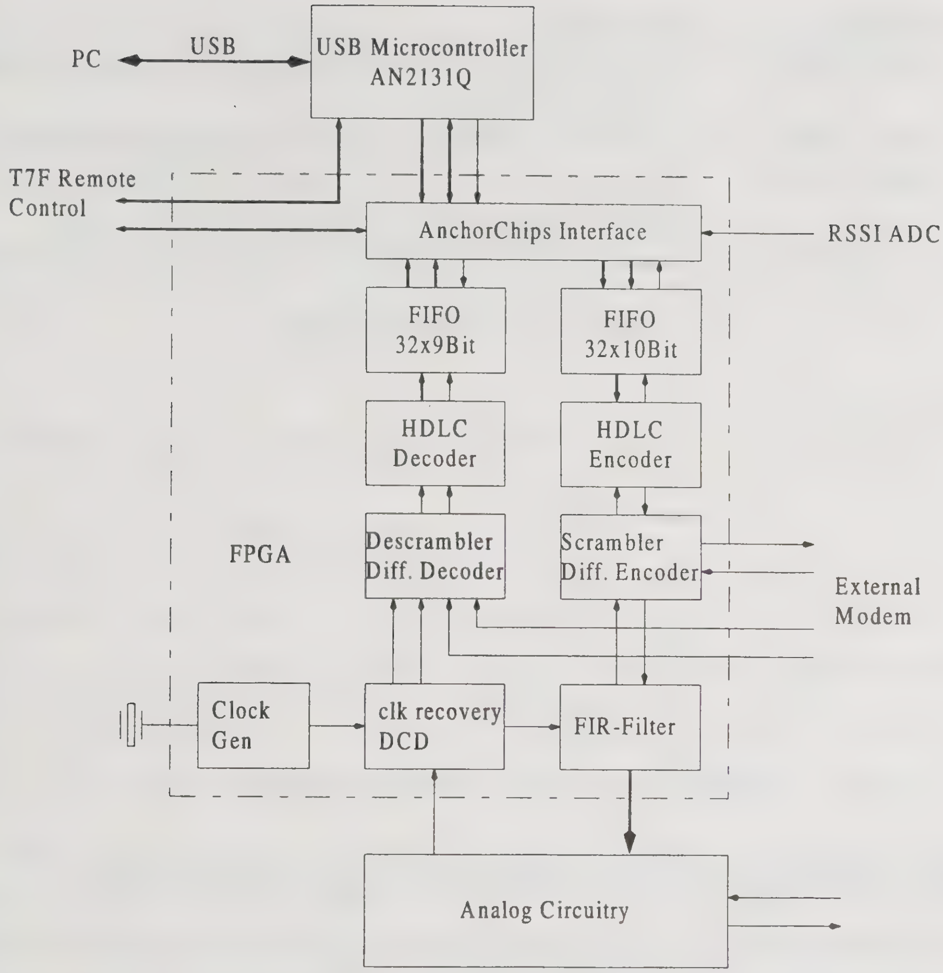



Figure 2: Diagram of the FSK mode

be either discarded or queued, and memory for a new one has to be allocated), the receive FIFO collapses multiple consecutive flags into a single flag.

The modem part of the FPGA is the same as in the EPP modem adapter, and a description may be found in [7, 8].

## 3.5 The AFSK Mode

Implementing FSK modems with FPGA's has been popular for quite some time, so there exist "competing" designs, most notably the YAM [11]. The flexibility of the FPGA's spurred interest in realising an AFSK modem with the same hardware, and YAM got its AFSK modem firmware about a year ago.

But because the analog circuitry of these modems was designed for the FSK case, the performance of the AFSK demodulators implemented on these modems is pretty unsatisfactory. The next section shows how a simple modification to the analog circuitry boosts AFSK performance tremendously.

| Tag (Address) | | Purpose |
|---|---|---|
| 00 | FSKTXDATA | Transmit payload data byte. The byte is fed through stuffing and CRC generator. |
| 01 | FSKTXCRC | Transmit 8 CRC bits. Data byte supplied is don't care. |
| 10 | FSKTXRAW | Transmit 8 raw bits, i.e. without feeding through stuffing and CRC generator. Can be used to transmit flags. |
| 11 | FSKTXRAWCLR | Like 10, additionally initializes the CRC register to all ones. |

Table 2: HDLC encoder tags

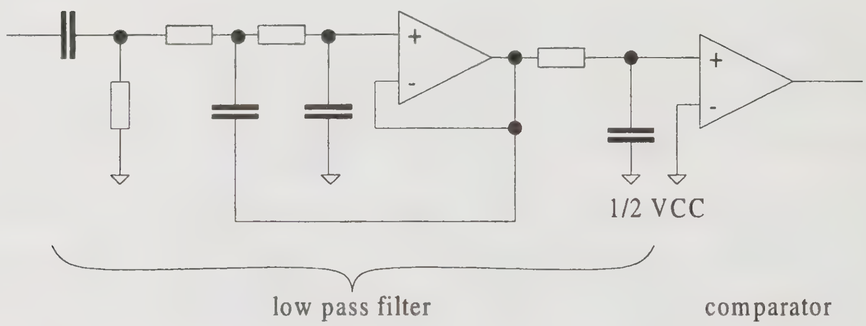



Figure 3: Standard FSK Modem Analog Circuitry

## 3.6 Modification of the analog circuitry for AFSK demodulation

Figure 3 shows a schematic circuit diagram of a standard analog circuitry of the receiver chain of a FSK modem. The comparator serves as the data slicer in the FSK case, but as a 1 bit A/D converter in the AFSK case. In literature, the signal to noise ratio of a $B$ bit A/D converter is given by $(6B+1.25)\ dB$. That formula however is based on the assumption that the quantisation error of two samples is uncorrelated, an assumption that is unrealistic for converters with a very low number of bits. This is also the reason why straight oversampling does not help.

So the AFSK demodulator has to work with a data stream that does not provide much information about the incoming signal. It therefore cannot make robust decisions.

### 3.6.1 ΔΣ A/D Converters

ΔΣ converters are very popular these days, even in quality audio A/D and D/A circuits, as they allow the construction of high quality converters with low precision analog components and some digital signal processing. Figure 4 shows a diagram of a ΔΣ A/D converter. The A/D and D/A converters do not need high precision, in fact they are often realised as 1 bit converters, as 1 bit converters cannot have a linearity error[3]. The only important aspect is that they are fast. The ratio of the sampling rate of the converters and

[3] it is always possible to perfectly fit a straight line through two points

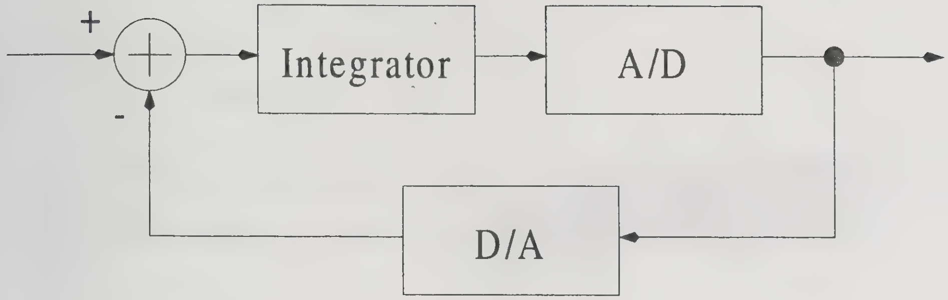



Figure 4: ΔΣ A/D converter diagram [12]

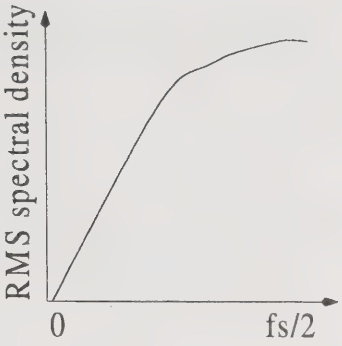



Figure 5: schematic plot of the quantisation noise

the highest signal frequency is called oversampling ratio.

The integrator's task is to change the spectrum of the quantisation noise and to decorrelate it. Instead of a flat noise spectrum, the integrator causes a reduction of the noise power at low frequencies and an increase in noise power at high frequencies. Figure 5 shows schematically the resulting quantisation noise spectrum. Since only low frequency signal components are of interest, a digital low pass filter removes the unwanted part of the spectrum and decimates the sampling rate to simplify the following circuits. The ΔΣ technique results in an improvement of 9dB or 1.5 bits per doubling of the oversampling ratio (using a first order loop [12]).

Comparing the diagrams of the ΔΣ A/D converter (figure 4) and the analog ciruitry of the FSK modem (figure 3) reveals that most components needed for a ΔΣ converter are already there. The comparator can serve as a 1 bit A/D converter, and the capacitor in front of the comparator can do the integrator's job. The only component missing is feedback part with the 1 bit D/A converter, which may be easily realised using a FPGA pin and a resistor (figure 6). In the FSK case, the FPGA pin may be programmed into high impedance state and therefore the resistor does not hurt.

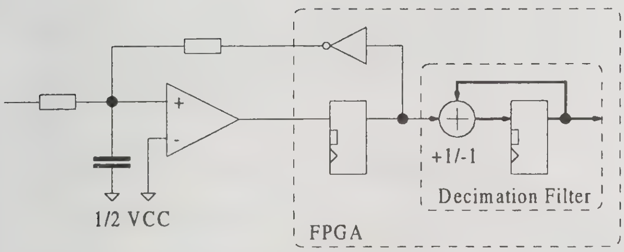



Figure 6: ΔΣ A/D Converter

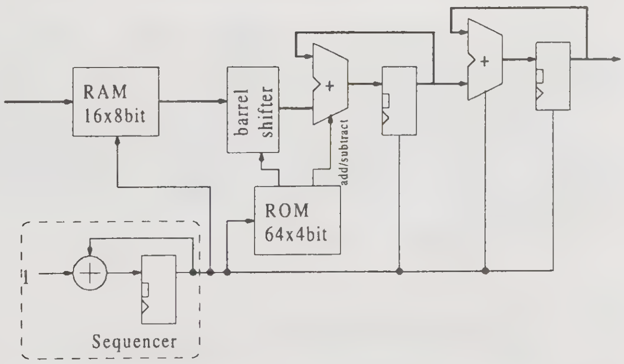



Figure 7: AFSK Demodulator

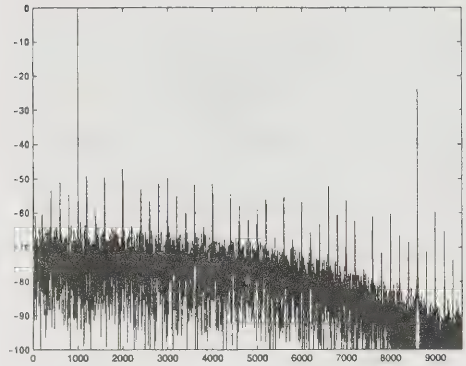



Figure 8: 'Loopback spectrum'

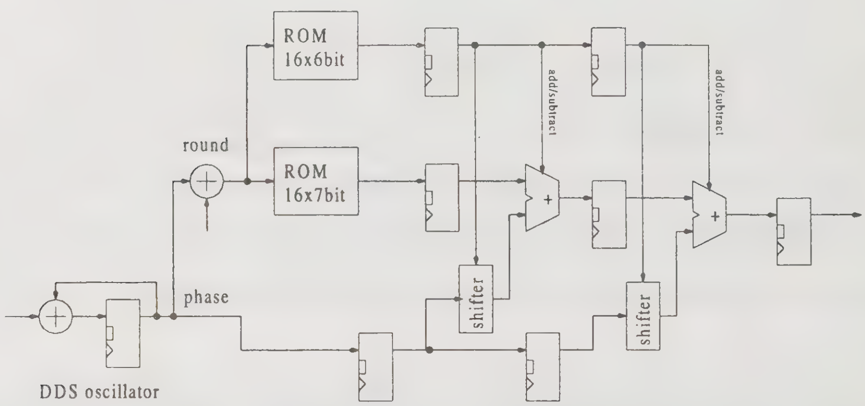



Figure 9: AFSK Modulator

### 3.6.2 Der ΔΣ-A/D-Wandler

Figure 6 shows a diagram of the A/D converter. The digital low pass decimation filter is realised as an "integrate and dump" or "boxcar filter". This filter has some strong sidelobes in the stop band, but these sidelobes are unproblematic and the filter is very easy to implement. Figure 8 shows the spectrum of the circuit in loopback configuration. The output signal of the A/D converter after the low pass filter was fed to the D/A converter in the transmit path. The circuit was excited with a 1kHz sine signal at the input, and the output was measured using a 16bit 48kSamples/s A/D converter. The signal was plotted using a $2^{16}$ point FFT.

## 3.7 The Demodulator

The demodulator block (figure 7) mainly computes four FIR filters. Simulation showed that the algorithm is very insensitive to quantisation errors of the FIR filter coefficients, therefore the coefficients have been coded with a sign bit and an exponent, replacing the multiplier by a barrel shifter. Both filter output value

pairs are combined by a sum-of-squares approximation and the resulting values compared to form the bit decision.

### 3.8 The Modulator

The mdoulator (figure 9) implements a sine generator. Because can only implement ROMs with very few address lines efficiently, the difficulty of this task lies in producing a good sine with only small tables. The circuit approximates $\sin(x+\Delta x) \approx \sin x + \Delta x \cos x$.

### 3.9 Microcontroller Interface

Since the number of gates for the AFSK modem is significantly bigger than for the FSK modem and the data rates are very low, the HDLC encoder and decoder have been moved to the microcontroller firmware. The FPGA only provides a doubly buffered shift register for the raw bits.

## 4 The Analog IO mode

Occasionally, it may be useful to operate the data radio as a normal voice FM transceiver. It is therefore useful to provide a mode where the transceiver baseband signal is sampled using the techniques presented in the previous section and forwarded to the host PC. The host PC then adapts sampling rates and forwards the signal to an installed soundcard for playback on a speaker. The reverse direction is similar. USB supports this mode of operation by offering isochronous transfers, which provide guaranteed bandwidth and latency, but do not correct errors by retransmission.

## 5 State of the Project

Prototype hardware is working since end of may, and beta firmware is working since end of june. The current prototype was designed and built in a hurry to get hardware fast to be able to start the software/FPGA firmware development. Therefore we will have a second and hopefully last prototype generation soon.

A user mode application exists that is able to transmit and receive frames and can be (cross)compiled for Linux and Windows. Under Linux it uses a small driver that exports USB host controller driver functionality to user mode using an ioctl interface, and under Windows it uses a generic driver provided by AnchorChips which does a similar thing.

Host operating system drivers do not exist yet. Under Linux, the USB host controller driver API is still emerging and not fixed yet, and under Windows the FlexNet AX.25 stack which we will be using is also undergoing some significant changes at the moment. We expect to have the drivers ready by fall 1999. The source code of the Linux driver will be available under the GNU GPL license in case anyone wants to port it to an operating system/AX.25 stack we will not be directly supporting.

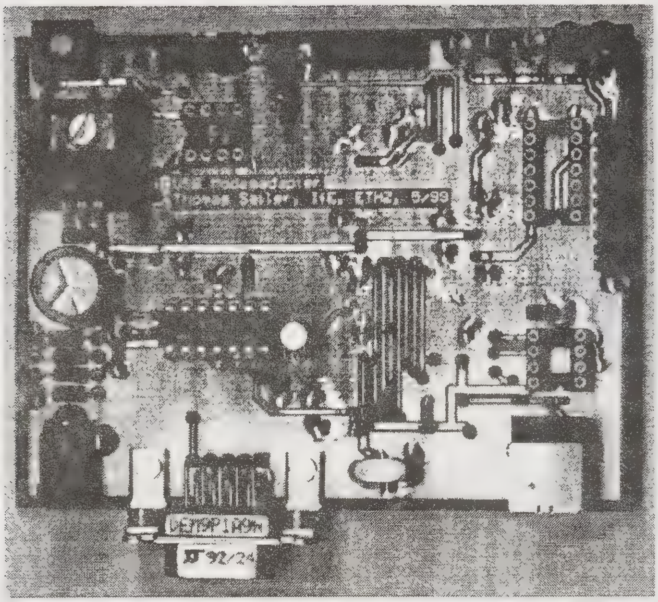


Figure 10: Top side of the prototype PCB

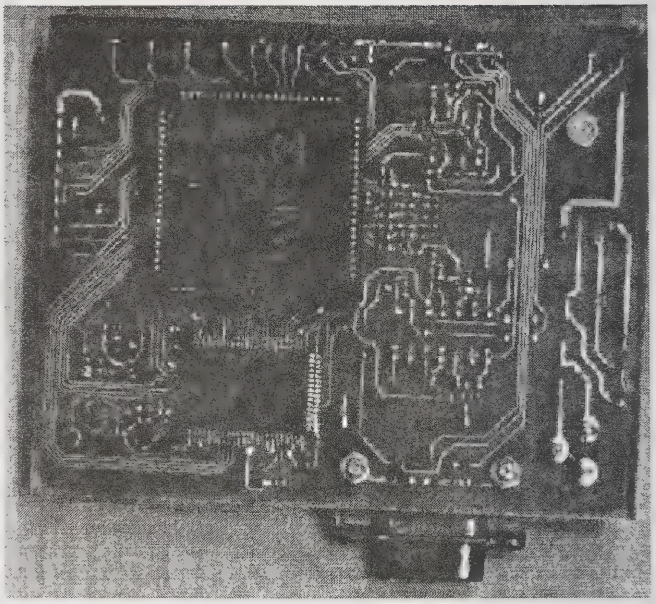


Figure 11: Bottom side of the prototype PCB

# 6 Conclusion

This paper presented a highly versatile amateur radio modem design that connects to a standard PC using the now ubiquitous Universal Serial Bus (USB).

The prototype was presented to the public at the Friedrichshafen, Germany Ham fair and generated significant interest from the amateur community.

The modem should be available soon from Baycom [5].

# 7 Outlook

Providing the interfaces to remotely controlling a transceiver is only the first step. We will be looking into combining modem and transceiver.

The hot spot of current amateur FSK transceivers is the modulator. Quartz oscillator transceivers often have deficiencies in the linearity of their voltage/frequency curve because they use a capacitance diode which has a very nonlinear voltage/capacitance curve. Synthesizer transceivers have a PLL with a loop filter whose cutoff frequency often lies in the middle of the signal band, requiring the injection of the modulation signal to the VCO and the reference oscillator. This method causes eye distortion even when the circuit was well trimmed.

Using a high bandwidth PLL and modulating it by modulating the divider of the prescaler using $\Delta\Sigma$ techniques looks like a simple yet promising technique to improve the modulator signal.

# References


[1] Xilinx Corporation *The Programmable Logic Data Book*
`http://www.xilinx.com`

[2] Wolf-Henning Rech, DF9IC, Johannes Kneip, DG3RBU, Gunter Jost, DK7WJ, und Thomas Sailer, HB9JNX, *Ein Modemadapter für den EPP*, 41. Weinheimer UKW-Tagung, Skriptum, 1996

[3] Johannes Kneip, DG3RBU, *Das FSK+–Modem mit Echoduplex*, Adacom Magazin 10, 1997

[4] Wolf-Henning Rech, DF9IC, *Modernes FSK-Modem – kompatibel zum Standard nach G3RUH*, Adacom Magazin 2, 1991

[5] Baycom ⊴ Hard- und Software GmbH, Bert-Brecht-Weg 28, D–30890 Barsinghausen, phone ++49 (5105) 585 050, fax ++49 (5105) 585 060, `http://www.baycom.org/`,
Email: `johannes@baycom.org`

[6] Martin Liebeck, DL2ZBN, und Alexander Kurpiers, DL8AAU,
*Hochgeschwindigkeits-Packet-Radio – Baugruppen für das 70cm Band*, Adacom Magazin 10, 1997

[7] Thomas Sailer, HB9JNX/AE4WA, and Johannes Kneip, DG3RBU, *An Inexpensive PC-Modem for 76.8kBit/s User Access*, 1998 ARRL and TAPR Digital Communications Conference, Chicago, Illinois

[8] Thomas Sailer, HB9JNX/AE4WA, und Johannes Kneip, DG3RBU, *Alternative Implementierung des EPP Adapters*, 17. Internationale Packet Radio Tagung, Darmstadt, 1998

[9] Thomas Sailer, HB9JNX/AE4WA, und Johannes Kneip, DG3RBU, *EPPFLEX – ein universeller Modemadapter für den EPP-Port*, 18. Internationale Packet Radio Tagung, Darmstadt, 1999

[10] Holger Eckardt, DF2FQ, *Handbuch zum 70cm-FM/FSK-Transceiver T7F*

[11] Nico Palermo, IV3NWV, *Yet Another 9k6 Modem*
`http://jupiter.web-hosting.com/~nicopal/yam/`

[12] James C. Candy, and Gabor C. Temes, *Oversampling Methods for A/D and D/A Conversion*, IEEE Circuits and Systems Society, 1992, ISBN 0-87942-285-8

# APRN™
# Automatic Picture Relay Network
# SSTV Picture Server

Keith Sproul, WU2Z, ksproul@vger.rutgers.edu
Douglas D. Quagliana, KA2UPW dqugliana@aol.com
Bob Bruninga, WB4APR, bruninga@nadn.navy.mil

**Abstract**

SSTV, Slow Scan Television, has been around for years. In the past, it was primarily used on HF. The equipment was big, bulky, and somewhat expensive. Nobody even thought of doing SSTV on VHF or portable. With the introduction of the Kenwood VC-H1 this stereotype has changed. This paper discuses software that takes the SSTV images sent from these portable SSTV systems and automatically puts them on a server that makes these pictures available to other Hams. This allows a Ham to send his pictures to a common site that other hams can recall the pictures from using touch-tone commands. This 'PICTURE SERVER' becomes a 'picture' repeater that also enables hams that do not have a good direct radio path to exchange pictures via this PICTURE NETWORK. In addition, the pictures can be viewed on a local network via Netscape or other web browser.

**Background**

Slow Scan TV was introduced in 1958. In the early days, the only way to receive SSTV was with home made hardware using long persistence phosphorus TV tubes. Then there were commercial units such as the ROBOT. Later there were software packages that ran on personal computers. One of these early programs could receive black and white SSTV images via the CASSETTE port on an Apple II. Later, there was a lot of SSTV activity on AMIGA computers. Now there are many programs that can receive SSTV pictures via a Sound Blaster card on a

PC. Mark Sproul, KB2ICI, wrote one for the Macintosh back in 1989, and there is MULTIMODE for the Macintosh that receives color SSTV pictures. There are several good programs for SSTV on Windows platforms including W95SSTV, ChomaPix, and others. In 1998, Kenwood introduced the VC-H1, Visual Communicator. The Kenwood VC-H1 is a truly portable unit for doing SSTV that changes the way of thinking of how SSTV can be used. Now, many people are wanting to take the VC-H1 and a handheld radio such as the Kenwood TH-D7A into the field and transmit pictures back to 'home base.' Fire companies are doing this on commercial bands, Emergency people are using this to show what is going on the field, and Skywarn people are using it to show the weather. It truly proves the saying "A picture is worth a thousand words."

**Operation**

This system provides for a mechanism where people with these new portable SSTV units (or any SSTV transmitter) can go out into the field and transmit pictures back to a common picture server. Then other hams can retrieve these pictures at their convenience. In addition to convenience, it provides the ability to get the picture to someone that is further away than what would otherwise be doable via direct simplex operation. It becomes a PICTURE REPEATER. The server also acts as a common storage and retrieval location for these pictures. The pictures can also be made available to a web server, thus allowing many people to see these pictures in near real time.

When you have a picture that you want to transmit, you simply transmit it on the appropriate frequency. The server recognizes the SSTV signal and receives it. When complete, it saves the picture to a file on the local hard disk and says via VOICE over the radio, "PICTURE ONE TWO FIVE RECEIVED BY WU2Z APRN." You now know that your picture was saved as picture 125, and you can tell your friends to go get picture 125.

In an emergency, special event, or weather situation, the APRN system would work as follows:

- Hams in the field would transmit pictures, each one getting a picture number assigned to it.
- Other Hams could retrieve those pictures by picture number.

• When one person requests a picture from the server, anyone listening can receive the picture, therefore cutting down on retransmissions.
• People at the command center would be able to see the pictures as they were received via a web browser.

By using common web browser software you eliminate the need to train people in the proper usage of specialized software AND you allow many people to view the pictures at the same time.

The expected scenario would be the hams at the 'EVENT' would transmit pictures and the officials back at the command center would be able to see what is going on as it is happening. Since the pictures all get date/time stamped, you would then have a real good history to go look at after the 'EVENT' was over.

Alternatively, imagine an 'EVENT' with a "see-in" in addition to a "talk-in." You could preload the server with images of maps, local landmarks, and roads and then instruct incoming participants to view the appropriate images as part of their directions to the 'EVENT.' A sophisticated server could even transmit a SSTV image of a map back the participant with "you are here" and "the event is there." In addition, since much of the functionality of the APRN SERVER is implemented in software, the server can be as simple as an HT and a laptop that you bring to the 'EVENT.'

**Software**

The software has three different components.

1: The SSTV receive software, running on a Windows or Macintosh computer to receive the pictures. This software also recognizes touch-tone signals for control. It also looks at data coming in from the Packet TNC via a serial port and will associate APRS packets with the picture if present. In early versions of this system, we used ChromaPix and our software together to accomplish this.

Later we wrote our own custom software that uses a common sound card and digital signal processing software running on the Pentium to handle the SSTV and touch-tones. The sound card samples the incoming audio at 11 kHz using

16 bits per sample, which is more than sufficient to accurately represent the incoming SSTV audio signal. The incoming audio is constantly being processed by the computer's microprocessor using digital signal processing to search for either touch-tones or a SSTV signal. No special slow scan television hardware is necessary for the APRN SERVER.

2: Specialized software to create HTML web pages showing the SSTV images. This program reads the collection of pictures created by the first program and creates an HTML page with thumbnails and text comments to display the pictures. In the final version, this is built into the receive software.

3: A normal WEB Server system.

The over-the-air commands understood by the APRN Picture Server are as follows:

| | |
|---|---|
| Rxxx | Recall Picture XXX where xxx is 010 – 999 |
| Dxxx | Delete picture XXX (this can be disabled) |
| Lxxx | Lock picture XXX so that it cannot be deleted.<br>It then has to be deleted manually. |
| Nxx | List (via voice) picture numbers new in the last XX hours. |

Pictures 000-009 can never be deleted and are reserved for the operator's ID screens and test patterns. Pushing 'R001' on your radio will recall a picture set up by the owner so that you know who owns and runs the Automatic Picture Radio Network system. Pictures 002 – 009 are test patterns. Picture 411 is on-screen help, and picture 911 tells you how to contact local authorities.

The server can be programmed to transmit this picture at a predetermined interval.

**Pictures shipped with the APRN Server Software.**

000.pcm APRN 'ABOUT' screen
001.pcm Owner/Organization information screen
002.pcm Second Owner/Organization information screen
003.pcm All Black test screen
004.pcm All White test screen
005.pcm Red Green Blue test screen
006.pcm Cyan Magenta Yellow test screen
007.pcm Gray Scale Test Screen
008.pcm Concentric Circle Test Screen
009.pcm Text test screen

411.pcm 'On-line' help screen.
611.pcm A screen that tells you who to contact if there are hardware problems

The *.pcm files contain the raw pulse coded modulation bytes, without any header information, suitable for sending directly to the sound card. A received slow scan television picture will be stored as a *.pcm file, in standard graphics file format or both.

**Hardware**

The hardware needed for this system is simple:

**<u>COMPUTER</u>**

100 MHz Pentium
Windows 95 or Windows 98
Sound blaster card
Serial Port
Internet connection (only required for Web Server connection)

**<u>RADIO</u>**
UHF or VHF radio

Unlike repeaters, this system will probably need to be monitored regularly so having it up on a mountain top, although desirable, might not be as convenient as you might need. Therefore, putting the APRN SERVER at someone's home or office that has a good location is desirable.

**Conclusion**

Slow Scan TV has taken on a whole new life. It is now being used as a PORTABLE system that can be taken on a hike, or to an emergency site and used in REAL TIME. This is an entirely new and different way to use an SSTV. This brings new life and new excitement to portable VHF/UHF operations. It also provides a new way to look at the world of amateur radio. We look forward to an entirely new aspect of Ham radio enabled by portable SSTV transmitters and fancy software to allow many people to participate.

**References**


http://aprn.rutgers.edu — APRN picture Server

http://dorm.rutgers.edu/~ksproul/sstv.html — List of SSTV Links

http://www.kenwood.net — Kenwood

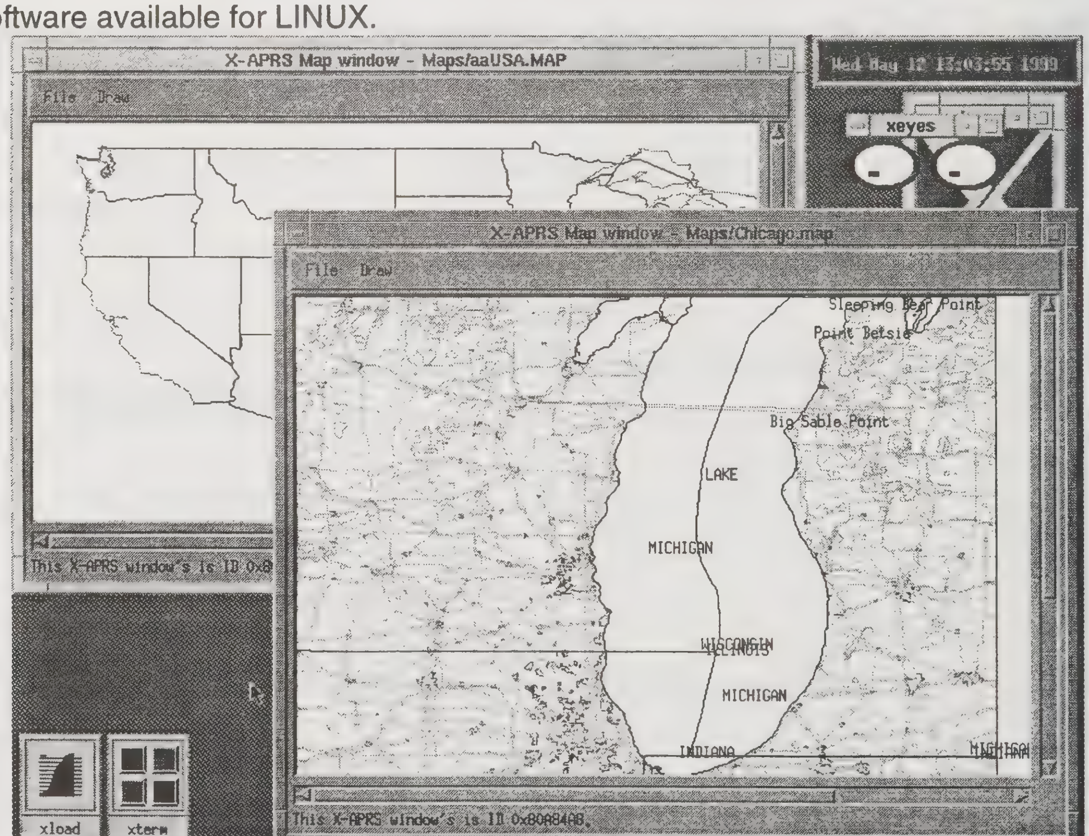
http://dorm.rutgers.edu/~ksproul/sstvhistory.html — History of SSTV (W9NTP)

# X-APRS™
# APRS® For X-Windows (Linux)

Mark Sproul, KB2ICI
1368 Noah Road
North Brunswick, NJ 08902
msproul@vger.rutgers.edu

**Abstract**

APRS, Automatic Position Reporting System, has been around for most of this decade. Part of the attraction of APRS is that it runs on many different platforms. There are versions that run on DOS, Macintosh, Windows, Palm Pilot, and now, The Sproul Brothers (TSB) have introduced a version that runs on LINUX, using X-Windows. This is an important addition to the APRS community because many Hams are starting to use LINUX and there are lots of other Ham Radio related software available for LINUX.



## Background

LINUX has become quite popular among Ham Radio people as an alternative operating system. There has been a large amount of work done for Linux for the Ham Radio community. Linux even has a system level Packet driver that speaks KISS to Packet TNCs. There has been a lot of pressure to get APRS running on Linux. This pressure has been building up over the last couple of years.

## Implementation

WinAPRS and MacAPRS are written using C/C++. The actual set of source code consists of 300 source files and just over 2 MILLION lines of code. The development system that we have been using for all of MacAPRS and WinAPRS is Codewarrior by Metrowerks. This is a full C/C++ development system that allows the programmer to compile programs for Windows on a Macintosh and to compile programs for Macintosh on Windows.  This system does NOT convert Mac programs to Windows or vise-versa.  The programmer must write the code for the specific operating system. This system just allows one program to have conditional compile statements that say IF COMPILING FOR WINDOWS DO IT THIS WAY, and IF COMPILING FOR MACINTOSH DO IT THIS OTHER WAY. Now that Codewarrior is available on LINUX, we can take the entire set of source code and add a third conditional to it of IF COMPILING FOR LINUX DO IT THIS THIRD WAY.

By using this ‘conditional compile’ method, we can have one large set of source code that is guaranteed to be compatible, and only have differences where the target operating system requires it.  For example, Protocol Parsing, which is the MOST important aspect of keeping the programs compatible, has nothing to do with which operating system it is running on, it is just straight ‘C’ code with no conditional compile statements at all.

Because of all of this, the conversion to a new platform becomes fairly easy. Once the primitives such as drawing lines and characters are implemented, the rest comes over fairly quickly and in large pieces. Another advantage is that all three versions become almost identical with all of the same features.

X-APRS is being developed for Redhat Linux. Eventually, we plan on making it work on some of the other popular versions of Linux and also SUN UNIX and other flavors of UNIX as needed.

**Conclusion**

X-APRS is progressing very nicely and is already in public beta testing at the time this article was written. There will be much more discussion of X-APRS at the Digital Communications Conference.

**Reference**

X-APRS is available at: ftp://aprs.rutgers.edu/pub/hamradio/APRS/

Documentation is at: http://aprs.rutgers.edu/

# APRS®
# Stand Alone Message Receiver

Mark Sproul, KB2ICI
1368 Noah Road
North Brunswick, NJ 08902
msproul@vger.rutgers.edu

Keith Sproul, WU2Z
698 Magnolia Road
North Brunswick, NJ 08902
ksproul@vger.rutgers.edu

**Abstract**

APRS, Automatic Position Reporting System, has evolved quit a lot over the last eight years. We can run APRS from our home, we can run APRS from our car. We can run with a computer in the car, or we can run with just a tracker unit. In this later case, we have stand-alone trackers, we have MIC-Es, PIC-Es and Kenwood Data Radios, In the case of MIC-Es, PIC-Es, and stand-alone trackers, the position data is being transmitted, but the received data is going nowhere. In these cases, for the most part it doesn't matter where the received data goes because we are not in a situation to use that data anyway. However, what if someone wanted to send you a message? If you have a MIC-E, PIC-E, or a stand-alone tracker, you just can't receive messages. The Kenwood TH-D7 can receive messages, but the other tracker units cannot. Wouldn't it be nice if you could receive messages even if you didn't have a computer hooked up in your vehicle. In addition, wouldn't it be nicer if we could receive the messages via voice?

**Background**

We (The Sproul Brothers) have Honda Goldwing motorcycles. These are big touring motorcycles that have a built-in CB radio and built-in AM/FM/Cassette system. They also have fully integrated intercom systems so that the driver and passenger can talk despite the high noise levels encountered at 65 mph on the highway. We have integrated Kenwood V7-A, dual-band radios with the VS-3 speech option into these motorcycles. We also have Radar detectors with

speech output. All of these audio sources feed into the audio system that feeds the speaker/microphone setup in the helmet. The driver and passenger can hear all of the different audio sources, they can talk to each other via the VOX activated intercom, and they can both talk on the CB or Ham radio via push-to-talk switches. There is switch that selects between the CB or Ham Radio. This switch not only switches which radio is hooked to the microphone and push-to-talk button, but it also switches the controls on the handle bar.

We have also added GPS units for navigation and for APRS tracking. We use either the Garmin GPS III+ or the Garmin Street Pilot. These GPS units and their maps are fully readable will riding the motorcycle. The APRS tracker uses the advanced data capabilities of the Kenwood TM-V7-A radio so that only one radio is needed for both voice and APRS tracking. We take advantage of the Voice output of the VS-3 option on the V7-A so that when we change settings on the radio, we get voice feed back from the radio. This makes it so we do not have to look at the radio display as closely.

What we would like is a unit that listens to the data coming in from the TNC, and if it sees an APRS message TO me, it will SPEAK the message into the audio system on the motorcycle. In addition to speaking messages that are to my specific call sign (WU2Z-10) I want it to speak any message to ANY of my other stations, WU2Z, WU2Z-3, etc. This would give me a hands-free method of receiving these messages safely. This not only applies to riding a motorcycle, but also when driving a car.

**Design**

If this project was being done on a normal desktop computer, there would be very little work involved. On the Macintosh, Text-to-Speech is built into the operating system and it only takes one line of code, literally, to do. On Windows, there is an AGENT that will do text to speech. The Windows version is much more complicated than one line of code, but is still quite simple to implement. MacAPRS has had message-to-speech for a long time and we will be implementing it on WinAPRS later on. When trying to design a system for text-to-speech from scratch, there is MUCH more work to be done.

You need a speech chip of some sorts, and you need a microprocessor with enough memory and speed to handle all of the involved processing. A normal PIC chip has the speed needed, but this project takes much more memory than the normal PIC chips. Recently, the PICs have become available in much larger configurations and this how we hope to implement this project. Steve Bible, N7HPR, from TAPR, has been helping us with the PIC aspects of this design.

The software in the PIC chip watches the incoming data and if it sees any messages to my call, it saves them and speaks the message. The system saves the 4 most recent messages to me, and allows me to replay them. This REPLAY feature is very important because if I can't hear it the first time, I need to be able to push a simple button and get it to play again. Pushing the button twice quickly, will replay the second one back, three times, the third, etc.

The main problem with this project is finding a good text-to-speech chip or phoneme-to-speech chip set. Given a good chip for the actual speech, the rest of this becomes fairly easy.

**Vocabulary**

Another reason that this project needs a reasonable amount of extra memory is all of the abbreviations used in Ham radio. Ham Radio has its own 'vocabulary'. For example, there are abbreviations of many common words used in ham radio. Many of these abbreviations trace their roots all the way back to the CW days. These abbreviations need to be taken into account and pronounced as a Ham would expect. The table below lists some of the common ones.

| | |
|---|---|
| CUL | See You Later |
| CQ | Needs to be pronounced as letters, 'C' 'Q' or SEEK YOU |
| FB | Fine Business |
| PSE | Please |
| QSL | Needs to be pronounced as letters 'Q' 'S' 'L' |
| TNX | Thanks |
| UR | You Are |
| YL | Young Lady |
| XYL | ex Young Lady |

## Conclusion

This could be a very useful addition to the APRS community. This unit and a MIC-E/PIC-E would make for a good communications system that could be left in your car all the time. With its message retrieval capability, you could also get messages that were received when you where not at your there.

Many modern radios have speech options available for them. Wouldn't it be nice if these message speaking capabilities were available as an option to put right into the radio?

## References


| | |
|---|---|
| http://www.microchip.com | PIC Web Site |
| http://www.tapr.org/tapr/html/pice.html | TAPR's PIC-E site |
| http://www.marc.org/ | MARC, Motorcycle Amateur Radio Club |
| http://msproul.rutgers.edu/Goldwing.html/ | Mark Sproul's Goldwing web site |

# Detailed Remote Weather Reporting VIA Packet Radio

Mark Wardell, N8LHG
1206 Burnt Pond Road
Ostrander, Ohio 43061
n8lhg@midohio.net

## Abstract

This paper will describe a unique way to view real-time dynamic weather station information via packet radio in extreme detail and accuracy.

## Introduction

Union County Ohio had been looking for an accurate and truly dynamic monitoring tool for watching weather station activity over a vast area for SkyWARN and EMA HAZMAT use. A program called URANIUM (For Radio Active Weather Reporting), gives Union County public services, as well as Amateur Radio SkyWARN people, a way to see actual moving or dynamic displays of up to 6 (six) weather stations on a single screen, and can collect data of an unlimited number (limited to actual channel use or congestion) of weather stations. Until now, this had never been done in a manner which is affordable to the Amateur Radio community. Watching the remote display of a Uranium station is the same as standing at the weather station itself watching the rose movement, and rise and falls of all data collected from each station. To date, Uranium is being used by assorted public services including SkyWARN, Ohio State Police, Union County Police, Fire, National Weather Services, EMA, as well as assorted Emergency Management EOCs (Emergency Operations Centers) in the state of Ohio.

Uranium makes use of a standard TAPR or TAPR clone TNC, 2 meter radio (or police scanner), and a computer. It monitors weather stations using a "type E" or "type F" beacon. These beacons are transmitted from each Uranium host station in a compressed unconnected info AX25 packet. One beacon is transmitted approximately once a minute. Each minute packet contains a wind speed and direction transmission, while every 5 (five) minutes, a beacon is transmitted which contains current temperature, average wind speeds and directions, humidity, high and low data for the day. Optional stations can transmit 1 (one) minute lightning strike data, and rain collection. The collected data is compiled at the remote site to display dynamic and real-time charting of all data, as well as an azimuth rose which cycles through the last 5 (five) received direction and speed beacons to give you a dynamic movement every second of each station.

The appearance is a real-time moving display of each weather host. With the beacons received, Uranium has the ability to display the heat index, dew point, cumulus cloud level based on collected data, heating and cooling degree units, average or mean temperature, as well as current trends. All this data can be archived and retrieved at later dates or times for charting, tabling, or whatever the needs may be.

Some of the other abilities of Uranium, which don't depend on the packet beacons include sunrise and sunset data, as well as moonrise and moonset data. Data displaying even the azimuth location of each

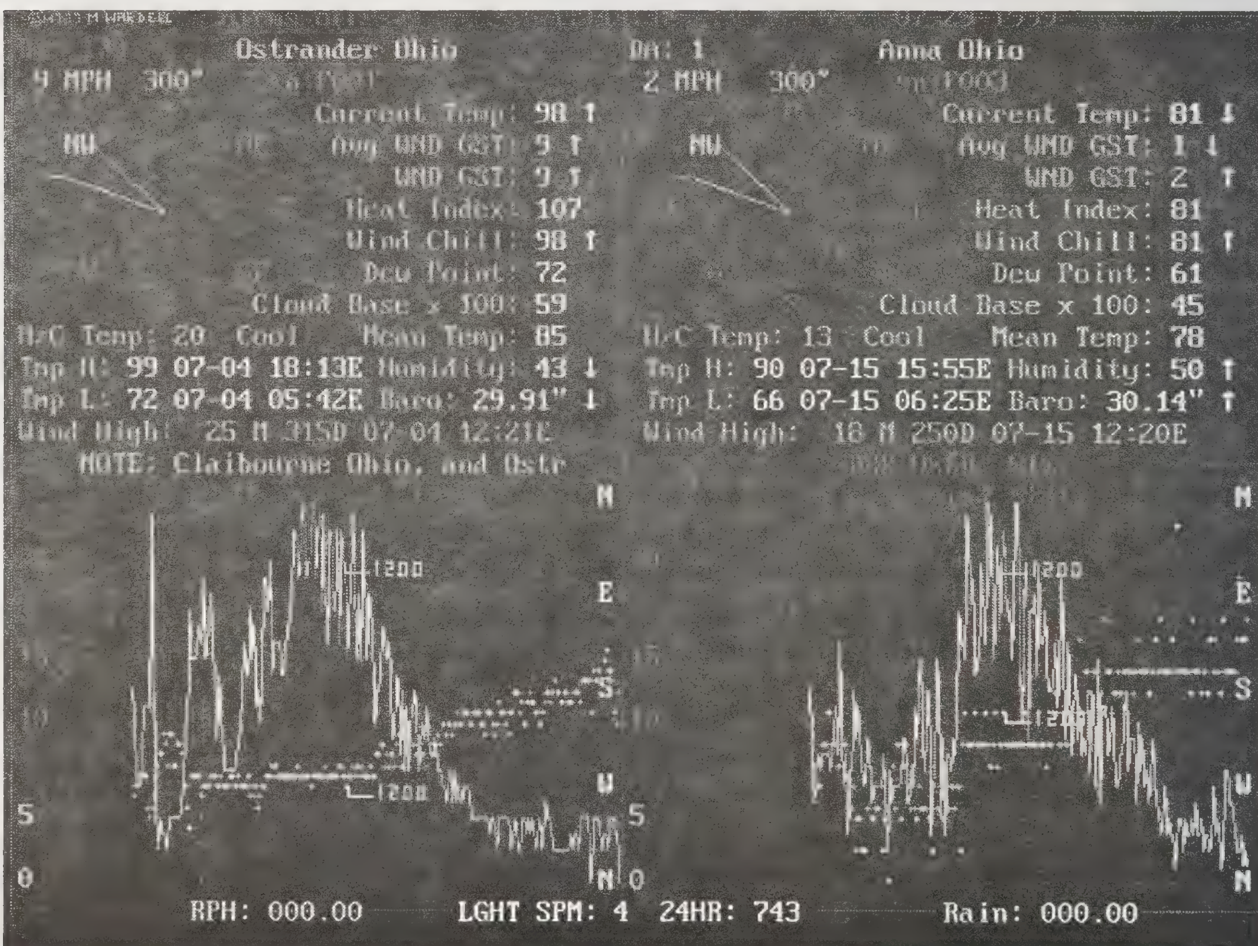
rise or set can be displayed as well as ecliptic moon information. WWV accurate time is also transmitted each hour to keep the remote stations accurate and correctly timed.

There is also a messaging ability within each host which can transmit Weather Forecasts or warnings. If the option is turned on, Uranium can update each remote with a message which is transmitted one line at a time. An entire Forecast message, if fairly long, can propagate to each remote station in about 10 (ten) minutes, keeping channel congestion to a minimum.

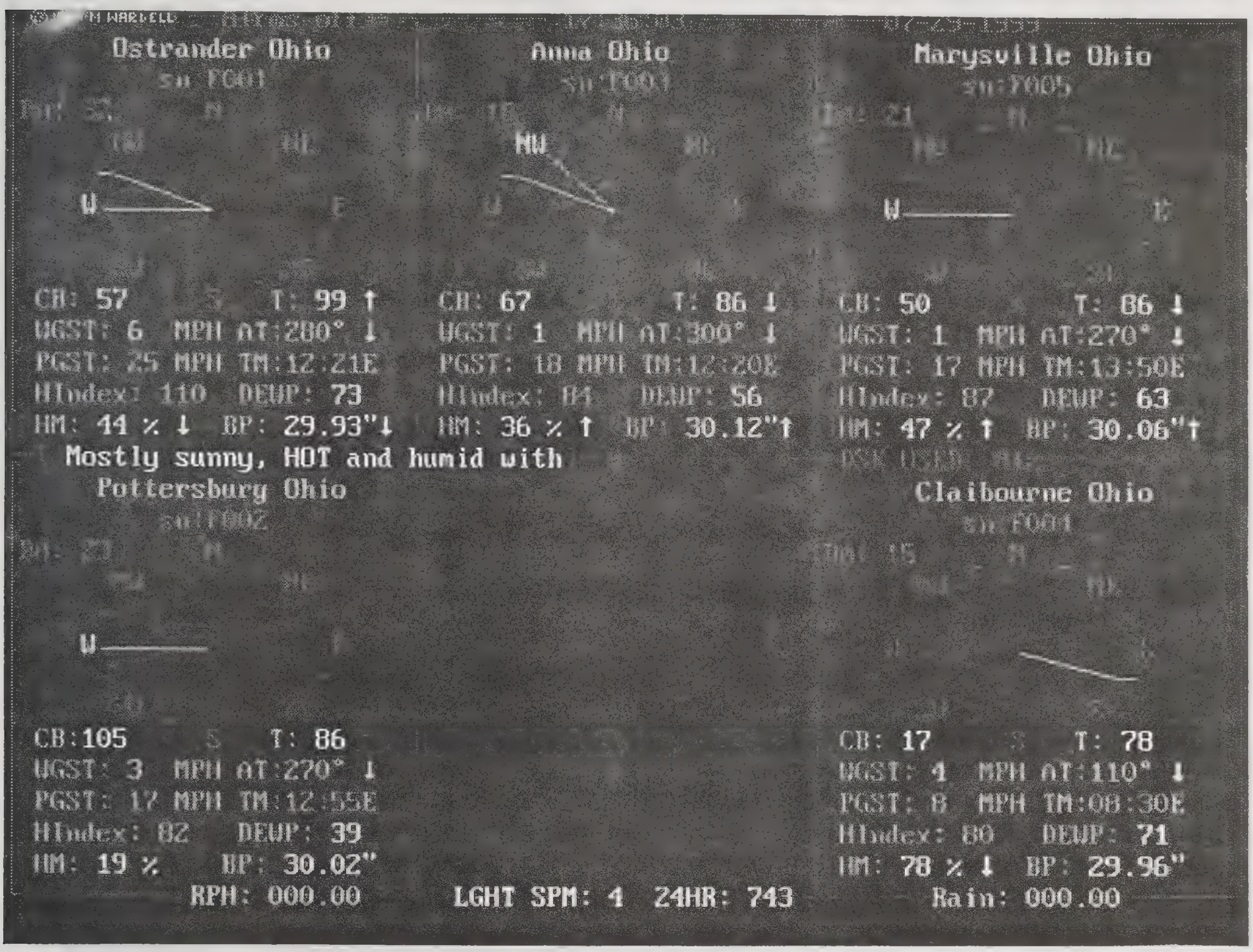
Uranium has a collection of alarm conditions which can be set to warn of any extreme change or conditions which the user wishes to be notified. These alarms can be audible or visual. Some of these include barometric pressure threshold alarms, wind gust threshold alarms, fog condition alarms, forecast warnings, watches, or advisory alarms.

The charting of the data collected is easy to read and intuitive to understand. For example, if a temperature drops below freezing, the chart line will change color to imply the threshold. All alarm conditions also can be set to chart changes.

The following example is one of the display screens showing wind speed and direction charting, as well as the above stations current displays all in real-time movement.



In order to monitor more stations, you can move to a 6 (six) station display which will display its current data dynamically moving.

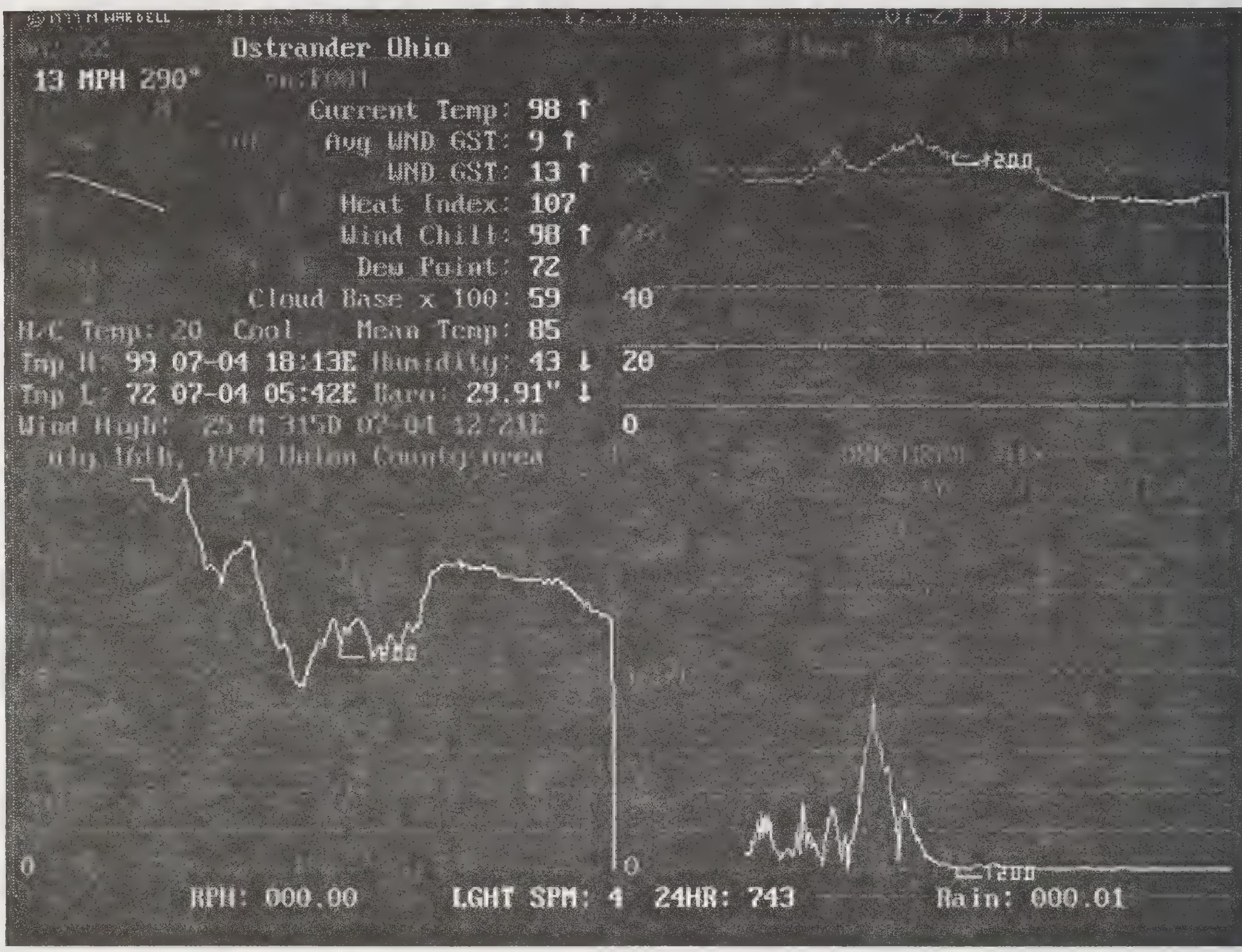



The above example shows 5 (five) stations being monitored. Note that there is a scrolling marquee in the center left of the display. This will scroll across the screen continuously. You can also see the current rainfall Rate Per Hour as well as the total rainfall since midnight. In the center bottom, you have a current count of lightning data collected. Lightning strikes per minute, and since midnight. Since Uranium is saving all its data, you can also see how much disk space you have remaining at the center right.

Lightning data can also be charted in real-time as well as the history. Lightning strike data is collected by an inexpensive device being sold by StormWise for about $60. This device can track lightning strikes as far as 250 miles from the receive antenna and is immune to man made noises.

Currently, Uranium hosts can transmit data from the EarthQuest weather stations, and the Capricorn II weather stations. These devices, recommended by the National Weather Service, can almost guarantee the accuracy of each display since they are both certified, and preset by the companies which sell the devices.

Below, is an example of single station monitoring. These charts include the temperature trend since midnight, barometric pressure, and lightning strike data received from the Ostrander Ohio weather station. Note, that even though this screen only shows one stations data, all other weather stations are still being monitored and data saved.




Time and space restraints don't allow me to go over every option built in to Uranium. But its abilities are impressive to the amateur meteorologist, as well as the professional.

Uranium was written by myself over the passed 3 (three) years with help from Meteorologists from the National Weather Service, EMA, and other public service people. It is extremely stable, intelligent, and has been known to run trouble free for months without a glitch. Uranium remote is available as Shareware, and can be distributed freely among licensed amateur radio operators. A small shareware fee can register and unlock all of Uranium's abilities. The only requirement is you need at least one weather host in your area to receive the weather data. If you have a Uranium host in your area, but don't want to use Uranium, all hosts transmit an APRS compatible beacon to show up on any current APRS station (assuming they are monitoring the correct frequency).

# Conclusion

In short, Uranium is an impressive weather forecast and monitoring tool for any amateur radio or public service / emergency use. It uses a standard packet station, and requires no special equipment on the receive side. Anyone who has a need for extreme weather detail, real-time weather station displays, or simply want to archive weather data, will find Uranium will fill those needs plus some.

Uranium has proven itself many times over in west central Ohio, and is a powerful severe weather warning and monitoring tool which evolves with its users needs and wants.

Uranium has no limitation on receive sites since what is being transmitted as an unconnected packet beacon.

Please feel free to contact me via email at **n8lhg@midohio.net** if you would like more information. I can also send you a copy of the users manual if you are interested in more detail as to how Uranium works, and its abilities.

73!

Mark, N8LHG

# APRS GENERIC DIGIPEATING SATELLITES for HT and MOBILE Satellite Communications

Bob Bruninga, WB4APR
115 Old Farm Court
Glen Burnie, MD 21060

It's time for mobile and handheld amateur satellite communications and we can do it easily. Proposals for amateur satellite constellations have been made in the past, but they assume a coordinated effort. Such an effort is unrealistic in the catch-as-catch-can amateur environment. This paper suggests that the future growth of amateur satellites can in fact accommodate uncoordinated growth and still provide synergistic advantages to mobile and handheld operations. There are several factors driving this concept making it easier, cheaper and more viable:

* Launches are plentiful, many free rides exist
* Electronics can be smaller than a softball
* The link from a handheld can be done with 1 watt
* Dual band HT's are plentiful
* One HT has a built-in 1200/9600 TNC and kbrd/display
* Mobiles are a convenient time to operate for many
* Travelers need satellite comms in remote areas
* HT/mobiles need little coax, no preamps, nor complex antennas
* Mobiles/Handhelds provide a common user configuration
* HT/omni downlinks are ideal for school demonstrations
* Many Schools want to build satellites for education
* Amateur Satellites should not compete with the free bandwidth of the internet for fixed stations. We should concentrate some of our satellite bandwidth on amateur communications for mobiles and handhelds.

In the past, HAMS were the driving force behind each of our satellites wanting to do something new and different and better for HAM radio. More bandwidth, higher speeds, new digital tecniques, longer ranges. The justification for a new experiment provided the rationale for the project. With that mindset, there were lots of new things to tinker with but little support or peer justification for flying just another one-of-the-same.

The current climate is quite different. Universities want to build satellites for the education of their aerospace students. There are at least a dozen such satellites currently under construction. In this case it is the "design and building" that are the main educational drivers. This means some of these projects are actually looking for a mission, or an excuse to build a satellite. These satellites become "amateur" satellites simply because the spectrum is easy to get. We, "the AMSAT community" should provide guidance so that we get what we want and need for the future of amateur radio while these programs with deep pockets get the educational experience they need.

**THE GENERAL MISSION AND BUILDERS CHANNEL:**

What we need is a published General Mission requirement and a Builders Channel. This will encourage builders to consider the inclusion of a general purpose FM or digital transponder on a common channel. The resulting simple transponders would all be similar and provide the amateur community with a fleet of general purpose relays for mobile and handheld communications while providing builders with a persistent mission requirement. Actually there are three missions and three builders channels, one for FM voice, one for brief 1200 baud digital position/status/message uplinks and one for bulk downlink of all amateur traffic using the very effecient PACSAT protocol at 9600 baud.

Focusing builders-in-search-of-a-misson on this General Mission and Builders Channel with only one power-class of users (mobiles and HTs) will make much more effecient use of our precious spectrum and also keep the remaining spectrum available for other analog, wide band and experimental modes.

**HT and MOBILE STATION CLASS:**

Even though HT's and mobiles differ in power by 10 dB (5 versus 50 watts), they are comparable in most respects. The downlink is identical. Both use omni receive antennas with no cable loss. The uplink is also quite comparable. HT's can easily vary their orientation and polarity and/or use small

The popularity of AO-27 is matched only by its own congestion. Imagine, however, if there were 12 other AO-27's on the same frequency giving two passes per hour round the clock *AND* a change in operating habits to let everyone have a chance. Photo by Phillip Fortenberry N5PF at last years AMSAT Conference

handheld gain antennas. The mobile is usually stuck with a vertical which has a null overhead and a good chance of being in the wrong polarity half the time. Thus the mobile's 50 watts is, on average, quite comparable to the handheld's 5 watts.. So, for the purpose of this paper, mobiles and handhelds are considered in the same power class. By having such a well defined user station baseline, it is easy to design a Handheld/Mobile mission for amateur satellites and to make good decisions about their operation on only a minimum set of builders channels as shown below.

**BUILDERS CHANNELS:**

The following builders channels are recomended for both digital and voice:

| | | |
|---|---|---|
| 2m | FM VOICE: | FM downlink to HT's and mobiles |
| 2m | 1200 Baud: | UI packet channel to HT's and mobiles |
| 2m | 9600 Baud: | Pacsat protocol message server channel |
| 70cm | FM Voice: | FM uplink from HT's and Mobiles |
| 70cm | 1200 Baud: | Alternate uplink from digital mobiles |
| 70cm | 9600 Baud: | Uplink for 9600 baud Message Servers |

These are single channels to be shared by any number of satellites built by anyone who wants to. The actual practical number is probably about a dozen or so since orbits with free rides will be random and we want to minimize overlaps. Actually, this is not a limitation as many of these payloads may be launched by the shuttle and other missions enroute to ISS and so will have life times on the order of a year or less. Brief periods of overlapping coverage will have minimum impact due to the 10 dB range variance, many dB of antenna pattern variance of both the satellite and users, and the 10 dB or so FM capture effect.

**PACSAT MESSAGE CHANNEL:**

Notice the provision for a 9600 baud Pacsat protocol downlink on 2m. This single channel with many birds has the potential to deliver to every ham on the planet with an HT, over 300k per pass per satellite. Or with 10 satellites in orbit, over 10 megabytes of personal messages and daily amateur radio news to everyone! This is where everyone monitors for their traffic. The 2 meter downlink is 9 dB better than any of the existing PACSAT 70 cm downlinks using the same power. Thus, easy to do with 1 watt microsats to HT's with rubber ducks. The channel also listens briefly for acks.

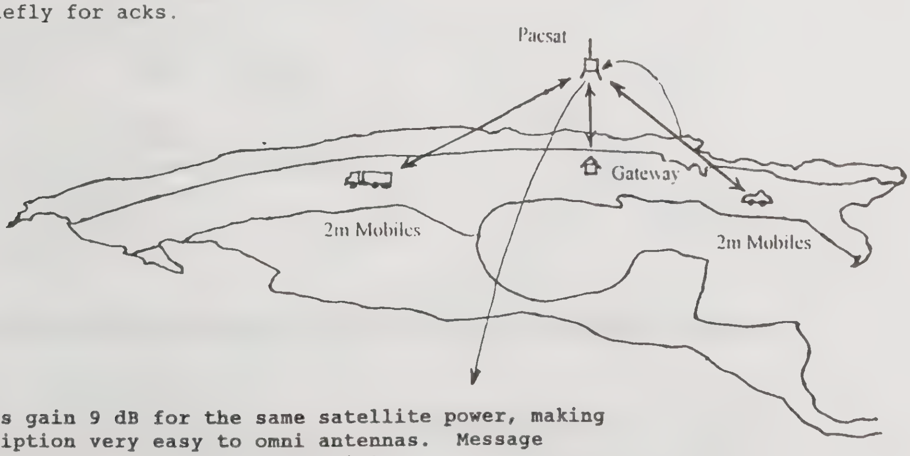
The objective of this channel is not to replace the existing PACSAT store-and-forward message/file systems, but to augment them for the downlink distribution of smaller real-time messages to all users usually in receive only mode. At 9600 baud, that is a tremendous asset for amateur radio. By using 2 meters, the lower path loss means that stations need not track the satellite. Just leave their packet capable HT on the 9600 baud channel and it will receive the traffic whenever a bird flies over even if they are in receive only mode.

Further, this PACSAT protocol downlink can be REGION specific by not putting the file system on orbit, but on the ground! The 9600 baud channel simply operates in bent-pipe mode for major internet linked file servers on the ground. Thus when over the USA, the channel is full of traffic for USA mobiles. While it is over Europe, the bird is supplying all European traffic. Thus, the channel can be optimized for each region. This greatly multiplies overall worldwide capacity.

Since each satellite is only serving as the on-orbit digipeater for regional message servers, each satellite is generic. Which ever one is in view at any time serves to deliver the messages to all mobile handhelds within 1500 miles. The Regional Message Server ground station continuously transmits a stream of all message traffic. New messages are initially repeated more often than older messages on a decaying schedule. For example:

| | |
|---|---|
| New messages are repeated | every 2 minutes |
| After 30 minutes | every 3 minutes |
| After 60 minutes | every 4 minutes |
| After 2 hours | every 5 minutes |
| After 4 hours | every 6 minutes |
| After 8 hours | every 7 minutes |
| After 16 hours | every 8 minutes |
| After 24 Hours | every 9 minutes |

Thus, at 9600 baud and an average message length of 1K, over 500 messages per pass per satellite can be delivered. Yes, it makes the satellite less valuable over the oceans and most of the 3rd world, but those areas are aptly covered by the existing Pacsats that cannot deliver traffic to handhelds.




**PACSATS with 2m downlinks gain 9 dB for the same satellite power, making handheld and mobile receiption very easy to omni antennas. Message delivery to/from anyone anywhere with an HT is possible.**

**The Kenwood THD7 HT can be a handheld satellite trasnceiver with buit-in 1200 and 9600 baud TNC's combined with keyboard and displays for position/status reporting and short messaging.**

DIGITAL MISSION:

Just like the many worldwide commercial satellite pagers and hand-held phones, Amateur radio now has its own "satellite handheld" in the Kenwood TH-D7. Just one packet can contain everything you need to know about a station and it takes less than 0.3 seconds per station. This means that 10 times more users per pass will be able to participate on the digital transponder as on the voice channel! In addition, the digital packets containing the stations position and status and messages can be interlinked into the worldwide APRServe internet backbone allowing worldwide communications between these handheld users anywhere to anywhere.

The digital mission using hand-held to handheld communications was recently demonstrated in an impromptu test during the first week in early June when more than 55 HT users easily communicated over 500 UI packets among 160 stations via the MIR digipeater. [ ] The presence of thousands of these shirt-pocket digital satellite communicators is an opportunity that should not be overlooked! Even conventional HT's can transmit APRS type position/status using only the match-box sized APRS Mic-Lite Mic-Encoder plugged into the mic jack.

The popularity and functionality of brief APRS UI type communications has been well dsmonstrated and is beyond the scope of this article. Readers can view the worldwide APRServe system live via TELNET to 199.227.86.221 port 23 at any time or check station positions via www.aprs.net. This system provides terrestrial HT coverage to over 90% of the US amateur population. (But less than 20% of the land mass and none of the oceans nor most foreign countries and Europe. Hence, the need to extend links via the digital builders channel).

FREQUENCIES:

One of the reasons for this builders channel concept is to make more effecient use of our existing spectrum and to allow for more orderly growth in the amateur satellite realm. There are many considerations in effectively designing a satellite link. First, 2 meters is, by an order of magnitude, the easiest band for handhelds, because the link budget is 9 dB easier and the doppler is negligible for existing FM radios. Conversly 70 cm has over 16 KHz of doppler and requires 9 times more power for the same performance. Thus, 70cm up and 2m downlink has the following advantages:

* 9 dB less power required on the satelite power system
* Omni spacecraft antennas require no attitude control
* No doppler tracking of downlink by users
* Compatible with all FM amateur radio equipment.
* Compatible with handheld TH-D7 packet HT

The UHF disadvantages are mitigated on the uplink as folllows:

* Ground users can use more power, or gain antennas
* User can compensate for doppler OR
* Satelite receiver can have AFC OR
* Satellite can use a wideband IF to minimize doppler

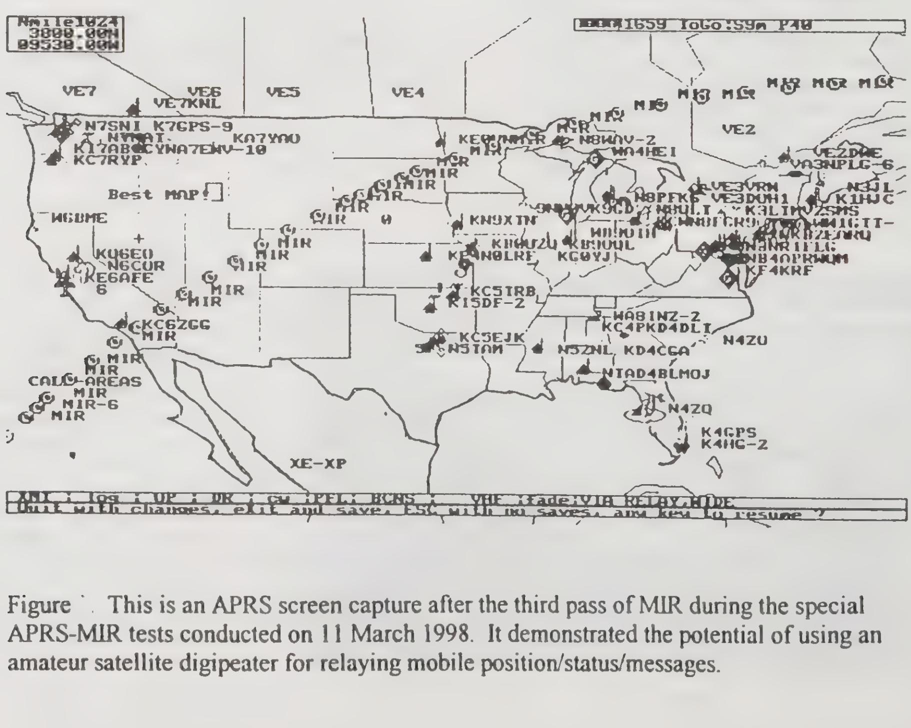


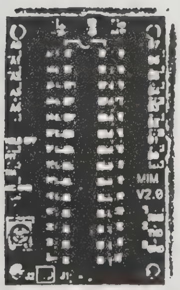

Figure . This is an APRS screen capture after the third pass of MIR during the special APRS-MIR tests conducted on 11 March 1998. It demonstrated the potential of using an amateur satellite digipeater for relaying mobile position/status/messages.

The only downside of this arrangement is the difficulty on the Voice birds of filtering to avoid 3rd harmonic desense of the satellite UHF receiver by its 2 meter downlink. On the 2m 1200 baud digital channel, HT-to-HT operations will usually use the satellite as a single channel digipeater for UI frames thus providing the 9 dB and zero-doppler advantages to everyone with an HT as was demonstrated via MIR. Simplex digipeating is required since the thousands of satellite packet capable HT's contain TNC's that can only operate on one band at a time. To help spread out the uplink load, the mobiles with 10 dB more power can transmit on 70cm, where the satellite can use AFC or a wideband receiver to mitigate the problem of high doppler.

Combining both uplink and downlink for the HT's on one channel is fortunately not a problem for the UI (APRS) application because dozens of ground stations are all linked by the internet. Thus only one ground station needs to receive the packet error free and it is distributed worldwide. Only 5 such ground stations would have a 99.99% chance of at least one of them hearing each packet and delivering it error free to the network even if their channel was more than 30% busy with local traffic.

In conclusion, I believe that establishing Builders Channels and published Common Mission Objectives for mobile/HT worldwide satellite communications could greatly benefit the Amateur Satellite community. It would provide a mission target for schools, a much improved use of or sparse satellite bandwidth on 2 meters and a constant source of satellites for a constellation of mobile communications amateur satellites. Further it defines a consistent user station class and power level for equal access by all, and promotes shorter practical protocols to allow more users brief access.


**The MIM/Mic-Lite module can be built into most Microphone enclosures or installed into a mini box to permit APRS osition/ status reporting from any HT.**




FULL SIZE!

# APRS® Mic-Lite™

Bob Bruninga, WB4APR
115 Old Farm Court
Glen Burnie, MD 21060

The Mic-Lite is the ultimate APRS Mic-Encoder for maximum flexibility and versatilty. It can be plugged into any radio in place of the microphone and gives you instant APRS position, telemetry and message reporting on any Radio. It is assembled from the MIM Module and a few off-the shelf Radio Shack components.

This Mic-Lite design solves many of the complications of the original APRS Mic-E for lightweight and portable applications. It solves the plethoria-of-mic-connectors problem by including an electret Microphone element and its own PTT within the Mic-Lite enclosure. Further it solves ground loop and power supply problems by operating from its own internal battery. Finally it solves the interface problem of combined PTT/Mic-Audio as used on most HT's. For most ICOM, Alinco, Yeasu and other HT's, only a single sub-mini phone plug is required. Kenwoods require two plugs and an external earphone.

Since the Mic-Lite is completely self-contained and runs on its own internal battery or Mic power, its best application is for special events where rapid configuration and portability are important. The internal 9v battery should last about 60 hours or over a month of typical commuting. The battery voltage is included in the telemetry packets so predicting battery replacement is easy to do. For mobile rigs, microphone power can be used instead of the internal battery.

ASSEMBLY: The Mic-Lite is a good evening construction project. Purchase the wired-and-tested MIM as indicated below, and then visit your local Radio Shack to pick up the remaining components for under $10 (you will pay slightly more since many components are packed 2 or more per package). The MIM module uses the same chip and firmware used in the TAPR Mic-E, and can be configured at any time in either the MIM or Mic-E mode. Since the bulk of the TAPR Mic-E is due to the front panel controls, Mic connectors and room for an internal GPS, this file shows how to use the basic MIM to make a much smaller and cheaper one called Mic-Lite(tm)

MIC-LITE: The full size TAPR Mic-E has two 16 position thumb wheel switches and a PERIOD pot to give the operator full control over Mic-E operations. For the Mic-Lite, most of these can be wired to small dip switches, replaced with single toggles or just jumpered to defaults with no switches. With no switches it can even be integrated inside the radio! Many routine applications of the Mic-E concept can be set to a default condition and front panel controls are unnecessary. Here is a table showing various levels of control.

| Front Panel | PATH Switch | MSG Switch | RATE | AUTO |
|---|---|---|---|---|
| Thumbwheels | 16 paths | 7 msgs | Pot 1-16 | On/Off |
| 10 pos Dip Sw | 16 paths | 7 msgs | Hi/Low | On/Off |
| Four switches | 2 paths | two msgs | Hi/low | On/Off |
| Three switches | 2 paths | One msg | Hi/low | ON/OFF |
| Two switches | one path | one msg | Hi/Low | ON/off |
| One switch | one path | one msg | Hi/Low | ON or OFF |
| No switches | one path | one msg | HI or LOW | ON or OFF |

I consider the Hi/Low to be the most important since it not only allows switching between "driving" and "parked", but also it triggers a "report now" whenever it is toggled. The next most important in some applications is the AUTO switch, since it allows you to operate in Mic-E mode on a voice repeater, or go to auto mode on 144.39 as a stand-alone tracker.

PACKAGING: There are five basic ways to package a Mic-Lite:

* IN-RADIO: This is easiest installation, but requires internal radio work and adding an external GPS jack. It also lacks front panel control which may or not be good.

* IN-MIC: This works well for mobile radios with room in their big Mic's when you can power the Mic-Lite from the mic power. You must add a GPS jack to the Mic enclosure and may want to add some buttons or switches. Wire the Hi/Low Period input to the UP button the Mic, and then any time you want to trigger a NOW packet, just press the UP (and then DN). The UP button still works as an UP button too! I got a MIM to fit in a very small Alinco TTONE mic! See also section on powering.

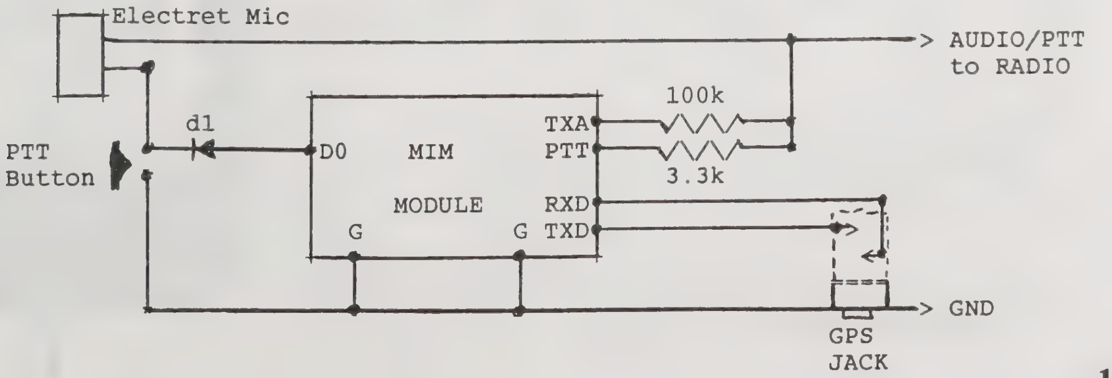
* IN-LINE: This either requires mic-in and out jacks so you can insert the Mic-Lite between the Mic and the Radio, or attaching it in parallel at the Mic connector on a dangel. This makes it easy to place anywhere for button access.

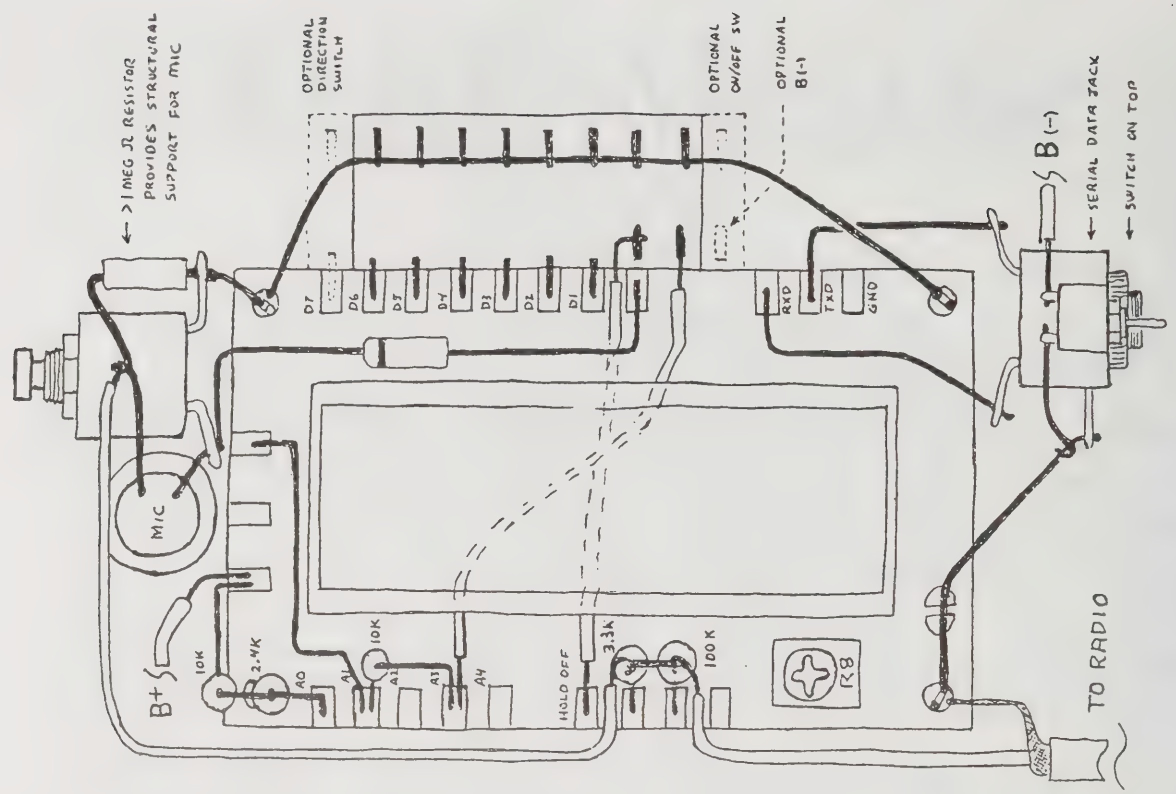
* NEW-MIC: This just adds an electret Mic and PTT to a small plastic case on which you can put any/all buttons you want. In effect, you are building your own Mic-E in your own Microphone case. Actually this is the most flexible option when included with a battery.

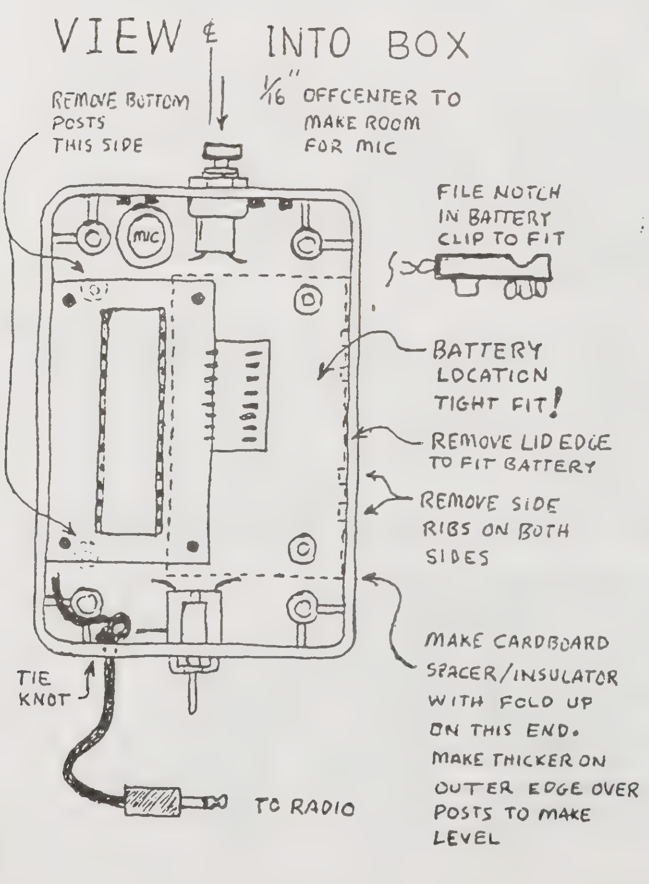
* FOR-HT: This arrangement is any of the above options, but uses its own internal battery so that it is trivial to plug into ANY radio (HT) with out any concern over power.

STAND-ALONE-MIC-E: This is the NEW-MIC and FOR-HT combination which I think is the most flexible. It builds the Mic-Lite into a small Mic enclosure with internal 9v battery so that it can plug into ANY radio or HT at the Mic Connector. The GPS then plugs into it. Here is the easy way to build this using the small plastic mini-box from Radio shack. Notice that it includes its own Electret Mic element to make it a completely self contained Mic-Encoder. Total parts cost at Radio Shack is $6.75 plus the MIM and Battery.

HARDWARE CONFIGURATION: The MIM is simply connected between the Microphone and the radio. For Handi-Talkies with combined PTT such as ICOMS, Yeasu Alinco's and most FRS radios, simply use a 3k and 100k resistor to join these two to a sub-mini phone plug as shown below.




>1 MEG Ω RESISTOR PROVIDES STRUCTURAL SUPPORT FOR MIC
OPTIONAL DIRECTION SWITCH
OPTIONAL ON/OFF SW
OPTIONAL B(-)
B(-)
SERIAL DATA JACK
SWITCH ON TOP
D7
D6
D5
D4
D3
D2
D1
RXD
TXD
GND
MIC
B+
10K
2.4K
A0
A1
10K
A2
A3
A4
HOLD OFF
3.3K
100K
R8
TO RADIO

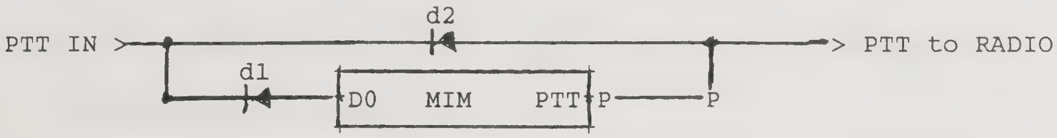

VIEW ℄ INTO BOX
1/16" OFFCENTER TO MAKE ROOM FOR MIC
REMOVE BOTTOM POSTS THIS SIDE
MIC
FILE NOTCH IN BATTERY CLIP TO FIT
BATTERY LOCATION TIGHT FIT!
REMOVE LID EDGE TO FIT BATTERY
REMOVE SIDE RIBS ON BOTH SIDES
MAKE CARDBOARD SPACER/INSULATOR WITH FOLD UP ON THIS END. MAKE THICKER ON OUTER EDGE OVER POSTS TO MAKE LEVEL
TIE KNOT
TO RADIO

FULL SIZE!

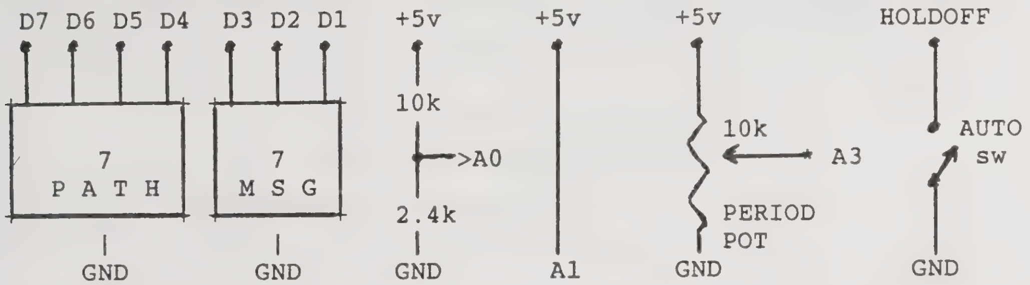

APRS®
Directional
Path R/W
Path 2
Path 1
Msg 4
Msg 2
Msg 1
Auto
Rate Lo/Hi
On/Off
Mic-Lite

GPS/SERIAL PORT CONNECTOR: Please consider using a 1/8th inch stereo mini jack for the GPS connector as shown. This is the same data connector used on the TAPR Mic-E, the PacComm Handi-Packet, and the Kenwood HT (though the Kenwood is a sub-mini 3/32, but the wiring is the same and only needs a RS adapter part # 274-397. TIP is DATA-OUT and RING is DATA-IN. Connect the GPS to DATA-IN.

CIRCUIT DESCRIPTION: Diode d1 lets the MIM sense PTT while also providing isolation to prevent the MIC-E from grounding the PTT lead when the MIC-E is turned off. The 100k audio resistor isolates the MIM so it does not load down normal Mic audio, and the 3k combines the DC PTT signal onto the Mic lead. For Kenwoods or other radios with separate PTT lines, then the following circuit is needed on the PTT circuit. Here diode d2 lets either the PTT-IN or the MIM key the radio. The points P--P show where to add a PTT LED if desired.




PATH and MESSAGE SWITCHES: Not shown in the schematic above are any of the optional PATH, MESSAGE and PERIOD connections. If unconnected, then the bits are considered a logic "0". Bits D2 to D8 connect the PATH and MESSAGE switches and A3 is the PERIOD adjust pot as follows:



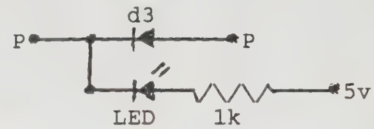

Note that A1 must be jumpered to +5 to enable the currently used DIGI mode instead of SSID mode. For the Mic-Lite, needing only to switch between R,W,W and W,W,W, simply ground D4 and D5 and add a select switch to D6. D7 is unused until we begin using directional and SSID routing some time in the future. A0 is connected to a 10k/2.4k voltage divider so that telemetry includes the battery voltage. A3 sets the PERIOD and the HOLDOFF pin may be connected to an AUTO switch, or GROUNDED.

MESSAGES: You can choose any number of switches and combination diodes to select any messages as follows:

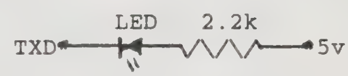
| | | |
|---|---|---|
| No switches | Off duty | |
| GND D1 | Enroute | |
| GND D2 | In Service | (Note AUTO PERIOD CIRCUIT BELOW) |
| GND D1&D2 | Returning | |
| GND D3 | Committed | |
| GND D3&D1 | Special | |
| GND D3&D2 | PRIORITY | |
| GND D3&D2&D1 | EMERGENCY! | Trips alarms & centers all maps to unit |

PERIOD POT: You can use a PERIOD POT or a 10k Pullup resistor and switch to control the "period" on input A3. With the switch open the +5 pullup will result in 16 times the programmed period. Grounding A3 will result in a POSIT-NOW and the pre-programmed rate. Or just hook to an existing Mic UP/DN button to serve as a POSIT-NOW button. See the section below on Automatic PERIOD adjustment between IGNITION on and IGNITION off.

PTT READY LED: You can add an LED to the PTT circuit to see when the Mic-E has a packet scheduled. You may want to talk a little longer and wait till a packet is ready before you release the PTT. To do this, add diode d3 to isolate the LED from the normal PTT circuit and connect at points "P" shown in the above PTT circuit.

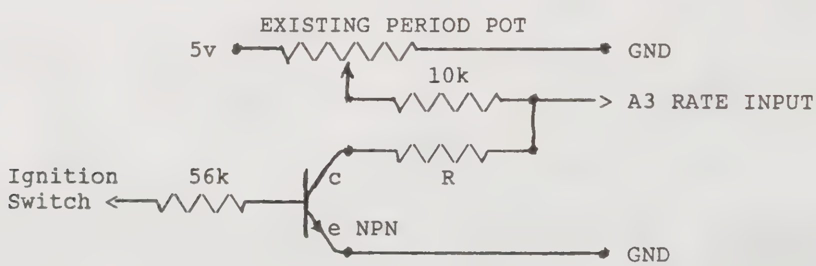



GPS ACTIVE LED: To see when the GPS has good RMC data, connect an LED to the TXD data line as follows:




BATTERY VOLTAGE TELEMETRY: Connect analog input pin A0 to a 10k/2.4k voltage divider so that your telemetry will contain your battery voltage. This combo makes 9v approximately equal to a telemetry value of 90. Or use a 5k pot in place of the 2.4k to tweak it exactly. Thus you can see the voltage of your battery to the tenth of a volt.

AUTOMATIC PERIOD ADJUSTMENT: Although the front panel PERIOD pot makes it easy to adjust your reporting rate, you can make it automatic between a short period when the engine is running and a long period when it is parked using the following circuit:




NOTE 1: The 10k resistor in series with the existing PERIOD pot input is there so that the new automatic circuit can share the input as shown.

NOTE 2: You can connect the collector to message bit D1 and this will change your message between OFF DUTY and IN SERVICE automatically!

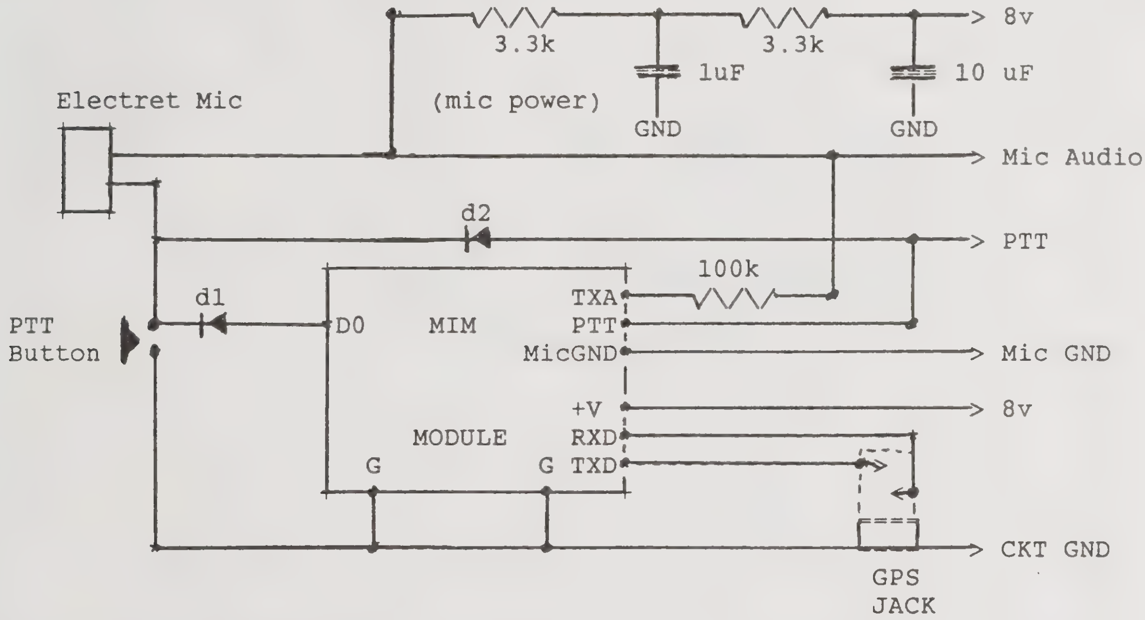
NOTE 3: Software program your Mic-E to a peak rate of 1 minute so the the 16:1 front panel PERIOD pot gives you a range of 1 to 16 minutes. The added transistor and new value R are switched in when the ignition switch is on. ALthough the front panel PERIOD pot can still override to any setting, set it normally to your desired max value, say 16. Now then you can choose R to give you:

PANEL POT will set the maximum period when the ignition is off.
R = 0 will always force the peak rate no matter where the pot is
R = 2k will force at least a MAX/8 rate
R = 4k will force at least a MAX/4 rate
R = 10k will force at least a MAX/2 rate

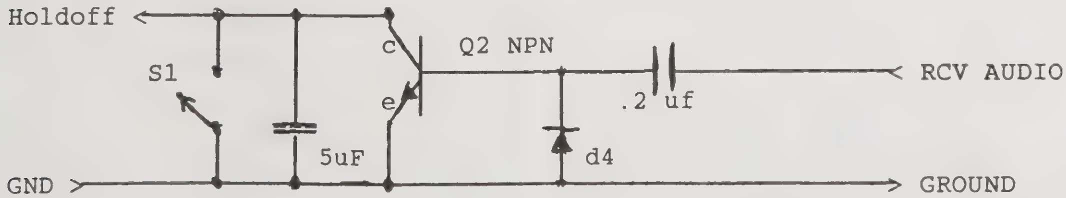
(These values are approximate, your milage will vary)

POWERING THE MIC-E: Although the Mic-E can be powered by the Mic power in some cases, the problems with alternator whine, GPS and processor digital noise on the power bus can be a big problem when interfacing to the mic connector of a mobile radio. If the GPS is also on internal batteries, then Microphone power should work fine. But if the GPS is externally powered by the vehicle battery, then you will surely have ground loop problems unless you are careful about grounding. The safest option is to use a 9v battery. It should last a month at 2 hours a day. See Mic-E.txt for other options. To minimize GPS noise, you may want to insert additional resistance in the GPS data line. See if 100K will work?

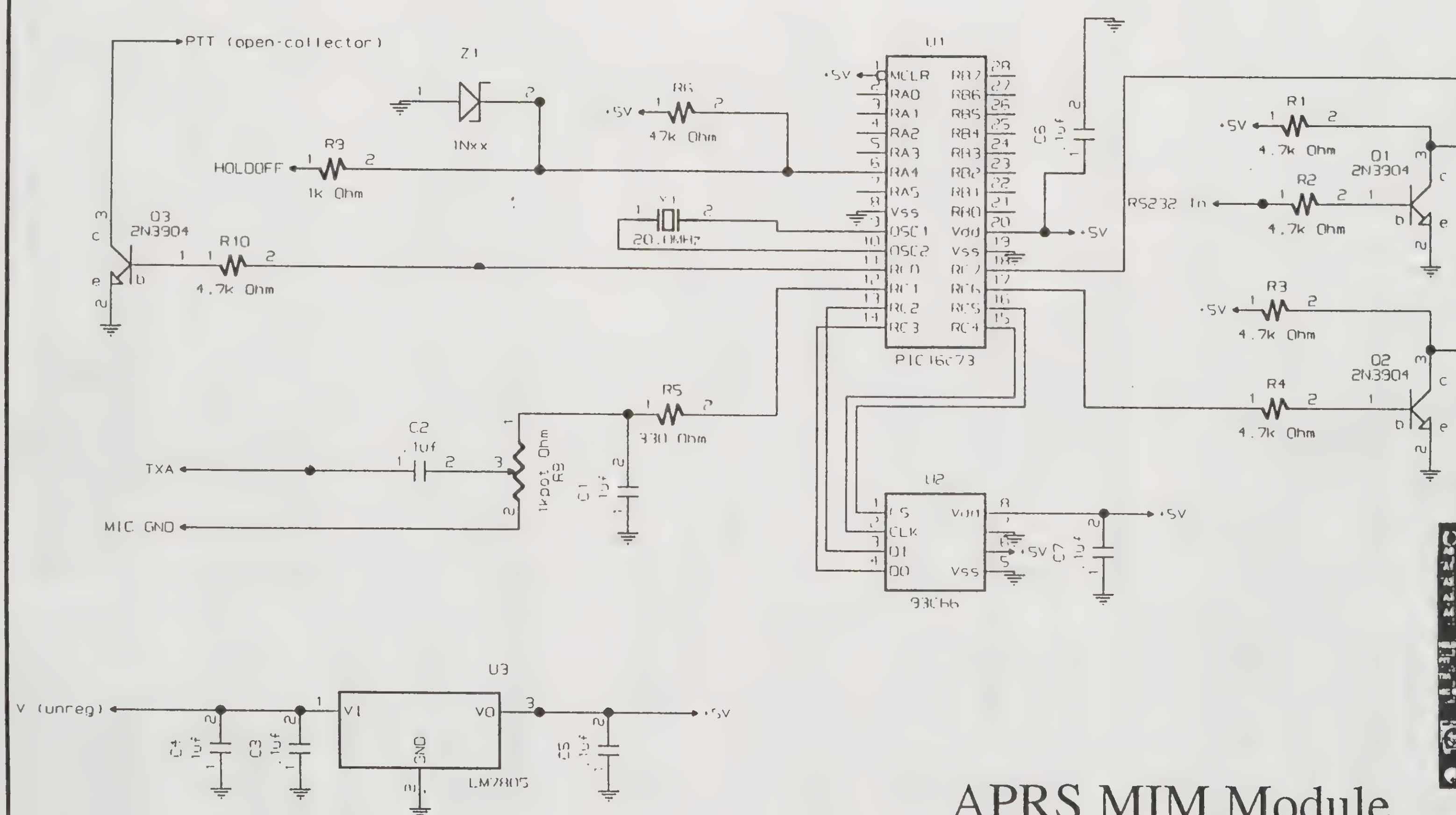
MICROPHONE-POWER: Note, in the simple circuit shown above, the electret mic element is also powered by the PTT voltage. If you build a Mic-Lite for any other microphone circuit, then you will need to provide a few volts of power for the mic element too. Add the two resistors and two caps in the circuit below to make sure the 8 volt microphone jack power (which is also powering the MIM) is clean enough for the mic:

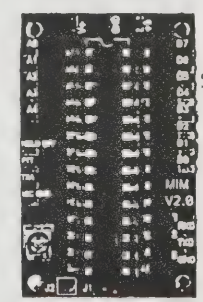



RECEIVER DETECTOR/AUTO MODE: If you want to operate in AUTO mode with carrier detect for collision avoidance (Not really needed for the Mic-Lite, since your EAR can tell you when to press PTT), add the following audio receive circuit to amplify the speaker audio high enough to trigger the hold off circuit. (This circuit has not been tested).




Q2, d4 and the two caps are an audio rectifier to drive the HOLDOFF input to the MIM when the radio is in use. This keeps the MIM from sending a packet. S1 is the AUTO switch. It can be used with or without this circuit. When it is shorted to ground, then this is the same as an "always busy" channel, so the Mic-Lite will never auto-initiate a packet. (It still will do the Mic-E function on release of PTT).




FULL SIZE!

# APRS MIM Module

Complete APRS Packet TNC transmit module. Transmits Position, BText Telemetry at user specified rate.

wb4apr@amsat.org

## Notes

**Notes**

Published by: ARRL The national association for AMATEUR RADIO